平成20年度

論文題目

FeliCaチップへの
秘密分散共有法の適用

神奈川大学 工学部 電気電子情報工学科

学籍番号 200502689

佐々木 賢

指 導 担 当 者　　木 下　宏 揚　教授

目 次

第 1 章　まえがき **5**

第 2 章　基礎知識 **6**
2.1 非接触 IC カードとは 6
2.1.1 RFID とは 6
2.1.2 接触式と非接触式の違い 7
2.2 FeliCa 概要 8
2.2.1 FeliCa の活用 9
2.2.2 FeliCa の暗号方式 10
2.2.3 FeliCa のファイルシステム 10
2.2.4 システムブロック・サービス 11
2.2.5 FeliCa のデータアクセスサービスの種類 12
2.3 秘密情報の分散 13
2.3.1 秘密分散を利用したサービス 13
2.3.2 (k,n) しきい値法 14
2.3.3 実現方法 16

第 3 章　秘密分散システムの概要 **17**
3.1 システム構築の準備と手順 18
3.1.1 開発環境 18
3.1.2 秘密分散システムの流れ 19
3.2 システムのモデル 20
3.3 システムの実行 23
3.3.1 実行テスト 23
3.3.2 テスト結果 23

第 4 章　評価 **29**

第 5 章　結論 **32**

謝辞 **33**

参考文献 **34**

付録 36 **質疑応答** 89

図 目 次

2.1 FeliCa メモリのフォーマット例 10
2.2 秘密分散共有法 14
2.3 秘密分散共有法 15

3.1 10 進整数「1234567」の保存形式 20
3.2 分散段階の処理図 21
3.3 再構成段階の処理図 22
3.4 分散処理（暗号）の実行テスト (1) 24
3.5 分散処理（暗号）の実行テスト (2) 25
3.6 再構成処理（復号）の実行テスト 26
3.7 各カードから読み取ったデータ 27

5.1 サンプルプログラムの GUI 38
5.2 サンプルプログラムの実行結果 39

表 目 次

3.1 開発環境 18
3.2 テスト内容 23
3.3 各カードの文字毎における分散結果 28

4.1 分散条件の違いによる変換後データ（容量）の比較 31

5.1 サンプルカードの読み取り条件 39

第1章　まえがき

近年、乗車券や電子マネーをはじめとして、社員証・学生証・会員証などといったカードで、読み取り面にかざすだけで認証・データの読み書きという処理が可能な非接触型 IC カード [3] の利用が多くなっている。これらのカードの内部は IC チップとアンテナで構成されており、リーダ/ライタと呼ばれるカードに読み書きができる装置にかざすと、装置から発せられる電波を受けて、カードはアンテナによって電力を得て IC チップが動作をする。IC チップには個々のカードを認識するために固有に記録されている ID や、利用者がデータを記録するためのメモリが組み込まれている。我々に最も身近な非接触型 IC カードの技術として「FeliCa」(フェリカ) [1] [9] [10] [17] がある。これを駅の改札を例に説明すると、IC カードは関東圏で利用されている「Suica」[14] や「PASMO」[15] といった IC 乗車券、リーダ/ライタは改札機である。利用者が改札機にカードをかざすと改札機から電波が発せられているため、カード内の IC チップが動作をして改札機にカード内に記録された情報を返す。カード内の情報は、この場合は乗降駅・残金などのデータであり、データを受け取った改札機はそれらの読み取ったデータを記録・処理するサーバーなどへ転送される。そのため、改札機としてのリーダ/ライタを制御する上位機器としてサーバーがあてられる。

ここまでは非接触 IC カードの簡単な利用方法を記述したが、これらはすべてカードという媒体を通して、個人およびその本人が有する情報をやり取りするための認証を行っているということである。認証という処理・動作は秘密情報が外部に漏えいするのを防ぐために認証情報（鍵・キーなど）によって所有者（本人）である確認を行う。通常、カードを利用した認証には１枚のカードのみであったり、暗証番号を入力するなど１人の管理者による認証が行われるが、本研究では、認証情報および秘密情報のそれ自体を複数の情報に分散して、管理者を複数にすることによって、任意に決められた２つ以上の情報が集まらなければ秘密情報を得ることができないという「秘密分散共有法」[2] [11] を非接触 IC カードに適用することを目的としている。

このシステムの利点として、秘密情報をある一人の管理者のみに預けるのではなく、一定の人数以上の承諾を得なければ秘密情報を閲覧することができない場合や、データの一部を紛失してしまった場合のバックアップ機能など、また管理者による情報漏えいや悪用を防ぐことが可能となる。

第2章　基礎知識

2.1　非接触IC カードとは

認証（認識・識別）の技術として現在私たちの身の回りには、商品を購入する際には“ バーコード ”から、代金を支払う際に“ クレジットカード ”で、また現金を ATM から引き出す際には“ キャッシュカード ”、万引き防止のために商品に“ タグ ”をつけて管理する、といった認証技術が多く利用されている。

この中でも、近年利用が増えているものに RFID[3] という技術がある。

2.1.1　RFID とは

RFID(Radio Frequency Identification) とは、電波を利用して人物や物品を自動的に認証する技術の総称である。これは、タグやラベル状に加工されたアンテナ付 IC チップを物や人に付与し、そこに記憶された情報をリーダ/ライタと呼ばれる装置で読み取ることで、物体や個人の認証を行おうとするものである。

最近では、RFID の持つ特性を情報システムに応用することにより、現在人の手により行われている多くの業務オペレーションを自動化し、あるいは簡素化することができるため、RFID を導入した企業は膨大なコストを削減できるといわれている。さらに人為的なミスの防止やシステムのリアルタイム性が向上することにより、情報の質が向上し、企業リソースの正確な把握や、迅速な意思決定を支援するものとしても期待されている。今では、人物や商品だけでなく、家畜の管理などにも利用されている。

RFID の利点として、無線を使用しているため読み取りが柔軟であり、現在の物体認証の主流であるバーコードや、近年携帯電話端末のカメラなどで多く利用されるようになってきた 2 次元バーコードなどにくらべ、読み取り面が必要ない。また、クレジットカードなどのような磁気カードといった接触が必要ないため、識別距離が長くなっている。その他に偽造がしにくく、安全性が高いという点もある。

2.1.2 接触式と非接触式の違い

RFID の技術が埋め込まれたカードを通称“ IC カード ”と呼ぶ。IC カードには、キャッシュカード大のプラスチック製カードに、メモリやマイクロプロセッサを内蔵した IC チップを搭載している。IC カードは、磁気ストライプカードと比べ、多くの情報を保存することができ、接触式 IC カードと非接触式 IC カードに区別される。

接触式 IC カード

接触式 IC カードには、カードの読み取り面に端子があり、それを読み取り装置の端子と接触させることにより、IC チップとの間でデータを読み書きすることができる。接触式 IC カードは、データを送受信する際に読み取り装置にカードを挿入し、接触電極をそれぞれ接触させなければならないため、摩耗や汚れなどから接触不良が発生する可能性があり、それを防止するためには定期的に点検を行う必要がある。また、カードの読み取り面は決まっているため、表裏、前後左右などを誤ると読み書きができなくなるため、利用者にとってはカードの取り扱いが煩わしい場面ができてしまう。

非接触式 IC カード

非接触 IC カードは、カード内部にアンテナの役割を果たすコイルを搭載しており、カードを読み取り装置にかざすことによってそこから発生している電波（磁界）の無線通信を利用してデータの送受信を行う。非接触 IC カードは、データの読み書きが可能な距離や通信方式の違いによって分類されている。距離の違いでは、密着型（〜2mm）、近接型（〜10mm）、近傍型（〜70mm）に分類される。また、通信方式の違いによって TypeA、TypeB、TypeC（FeliCa）に分類される。

2.2 FeliCa 概要

非接触型 IC カードには、2.1.2 で述べたように通信方式の違いによってカードタイプが分類されている。本研究では、システムを構築する非接触型 IC カードとして現在私たちが最も多く利用し身近なものであろう、IC 乗車券（関東圏では Suica、PASMO）に採用されている方式である“ FeliCa（フェリカ）”[1] を用いることにした。

FeliCa はソニー　が開発・商標登録した方式であり、「felicity:至福」から名前が由来している。当初は非接触 IC カード (ISO 14443)TypeC として採用される予定であったが、その計画は最終的に非接触型 IC カードとしては世界で初めて、セキュリティ評価基準の国際標準である ISO/IEC 15408 EAL4 の認定を受けた。

非接触型 IC カードの規格には様々な種類があるが、FeliCa には、次のような仕様や活用方法・利点がある。[4]

2.2.1 FeliCa の活用

電子チケット・乗車券

FeliCa は瞬時にデータを読み取って処理することが可能である。よって、利用金額の追加 (チャージ) も簡単に行え、繰り返し利用することが可能である。そのため乗車券への応用に非常に適しており、大手鉄道会社の IC カード乗車券に採用され、定期券やプリペイドカードとして利用者が増えている。さらに、さまざまな地域においてのバス乗車券としての運用も広がりつつある。

電子マネー

FeliCa 技術を採用した電子マネー“ Edy [1] ”の利用が進んでおり、このマークのある加盟店であれば全国で利用が可能で、大手コンビニエンスストアや大規模複合施設でのショッピングを始め、自動販売機などでも利用することができます。また、レジ処理は店側にとっても快適な処理となる。利用金額のチャージには店舗などの入金機だけでなく、インターネット上からも簡単に行うことができる。

社員証・学生証・会員証

出退勤管理やオフィス・事業所、研究室などの入退室管理、個人認証などに利用することができる。[6] [7] [8] また、電子マネーの機能と併用することにより、社内・学内の売店・自動販売機・食堂などでの買い物に利用することが可能である。

携帯電話端末への搭載

カード内に搭載されている FeliCa の IC チップを携帯電話端末内に搭載したものをモバイル FeliCa と呼ぶ。これによって、普段持ち歩いている携帯電話端末を FeliCa の非接触型 IC カードと同様のサービスを利用できるほか、FeliCa チップとのデータ送受信を利用したアプリケーションサービスの利用が可能となる。代表的なサービスとして“ おサイフケータイ ”があり、電子マネーとして利用するほか、残高状況を携帯アプリによって閲覧することができる。

[1] ビットワレット株式会社のプリペイド型電子マネーサービス

2.2.2　FeliCa の暗号方式

FeliCa がもつ暗号処理として、通信路に DES [2] もしくはトリプル DES [3]、相互認証にトリプル DES を採用している。Dual カードタイプ (接触/非接触兼用) では公開鍵暗号方式の処理が可能なものがある。

1 枚のカード (1 つのチップ) に複数のサービス（IC カード乗車券、電子マネー、社員証など）を搭載可能であるが、サービス利用時には、個々のサービスごとにアクセス鍵（共通鍵）を使って相互認証を行うのではなく、複数のアクセス鍵を暗号化し合成させた“ 縮退鍵 ”を用いて、一度に最大１６のサービスを相互認証することが可能である。縮退された鍵から元の鍵は生成できないことから、セキュリティレベルを落とすことなく処理速度の高速化を行っている。

2.2.3　FeliCa のファイルシステム

FeliCa チップ内におけるメモリのフォーマット例を以下の図 2.1 に示す。
なお実際のフォーマットはカードを発行する事業者によって異なっている。

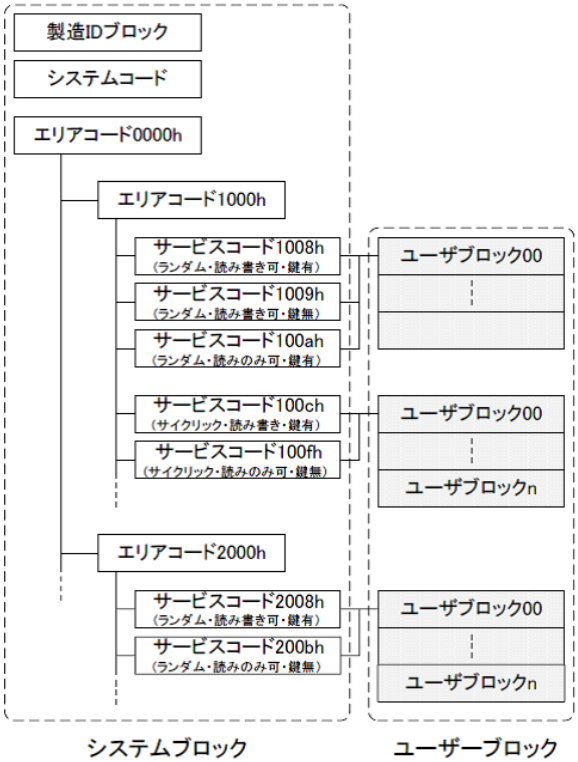


図 2.1: FeliCa メモリのフォーマット例

[2] IBM 社によって開発された秘密鍵暗号化アルゴリズム。
[3] DES を三重に適用するようにした方式のこと。コンピュータの性能向上に伴って DES 暗号を解読される危険性が高まったため、同じ方式を三重にかけることにより、強度を高めた。

2.2.4 システムブロック・サービス

ブロック

FeliCa 内のメモリ管理は 16 バイト毎に行われている。この 16 バイトの単位をブロックと呼ぶ。ユーザーがメモリにアクセスする際は、エリアコード・サービスコードと呼ばれる 2 バイトのコードを使用する。

また、FeliCa 内のメモリはユーザーデータが書き込まれるユーザーブロックと、FeliCa の構成情報が保存されているシステムブロックと呼ばれる領域に分けて管理されている。

製造 ID ブロック (製造 ID、製造パラメータ)

製造 IC ブロックの構成は、製造 ID(IDm) と製造パラメータ (PMm) のそれぞれ 8 バイトで構成されている。IDm は FeliCa を識別するために固有の番号が記録されており、認証などに利用することができる。PMm は製品のバージョン情報などが記録されている。これらの値はカード発行時に登録され、書き換えることができない。

ユーザーブロック

ユーザーブロックは、ユーザーデータが書き込まれる領域であるが、直接読み書きすることはできない。アクセスにはユーザーブロックに関連付けられたサービスを使って間接的に行われる。

システムコード

システムコードはカード発行者を識別するためのコード (2 バイト) である。

エリアコード・サービスコード

エリアコード (2 バイト) は、使用可能なサービスコードの範囲を示し、ユーザーブロック数などを設定コードである。(Windows のフォルダに該当する)

サービスコード (2 バイト) は、ユーザーブロックへアクセスするためのコードである。(Windows ではファイルへのショートカット名に該当する)

サービスコードによって以下のような属性が定義されている。

- サービスの種類 (ランダムサービス、サイクリックサービス、パースサービスの 3 種類があり、Windows では拡張子に該当する)
- ユーザーブロックに対するアクセス権 (読み書き可/読み込みのみ可、また、読み書き時の鍵の有無を設定する)

2.2.5 FeliCa のデータアクセスサービスの種類

ランダムサービスタイプ

ランダムサービスは、ブロック単位で自由にデータを書き込みできる汎用的なサービスである。属性は以下の 4 通りを持たせることができる。

- リード/ライトアクセス (セキュリティ認証必要)
- リード/ライトアクセス (認証不要)
- リードオンリアクセス (セキュリティ認証必要)
- リードオンリアクセス (認証不要)

サイクリックサービスタイプ

サイクリックサービスは、ブロック単位で任意のデータを読み書きできるサービスである。ユーザーブロックの中で、書き込みが行われた順番が記録されており、常に未書き込みブロック、または一番古いブロックに対して書き込みを行う。そのため、ブロックを指定して読み込むことは可能であるが、ブロックを指定して書き込むことはできない。属性は以下の 4 通りを持たせることができる。

- リード/ライトアクセス (セキュリティ認証必要)
- リード/ライトアクセス (認証不要)
- リードオンリアクセス (セキュリティ認証必要)
- リードオンリアクセス (認証不要)

パースサービスタイプ

パースサービスは、料金徴収などを想定して減算する機能を付加したサービスである。

2.3 秘密情報の分散

秘密情報を紛失などから守るためには、そのコピーを作って複数の場所に保存することが望ましが、コピーの数を多くすると盗難の危険が増大してしまう。一方、コピーの数を少なくすると、すべてを紛失してしまう危険が増大する。この相矛盾する二つの問題を解決する方法が“ 秘密分散共有法 ”[2] である。

2.3.1 秘密分散を利用したサービス

秘密分散を利用したサービスとして HITACHI が開発した“ 電子割符 ”[12] と呼ばれるものがある。そもそも割符とは従来からある考えであり、

1. 木片・竹片・紙片などに文字・図・絵を記し複数に分割する。
（ひとつひとつでは意味をなさない。）
↓
2. 全ての符が組み合わされる事により、始めて元の形に復元される。
（１つでも欠けたら復元できない。）

というものである。

この割符の機能をパソコンのデータに常時適用して、セキュアなデータ管理を実現するサービスがある。このサービスは「USB フラッシュメモリ」「電子割符クライアント」から構成されるシステムであり、以下のような特徴がある。

- 常に分割して管理するため「盗難・紛失」に強い
設定したフォルダの配下のファイルは、常に PC のローカルドライブと専用 USB フラッシュメモリに割符化（分散）された状態で格納される。盗難・紛失しても割符ファイルから、内容を推測することはできない。
- セキュアな USB フラッシュメモリにより情報を二重保護
専用のツールを使ってログインしないと、専用 USB フラッシュメモリ内のデータが見られない。万が一、専用 USB フラッシュメモリを紛失しても、第三者に参照されないため安心である。
- 専用画面によるファイル管理
割符ファイルを通常ファイルのように一覧表示する専用画面により、ユーザにファイル参照時やファイル登録時の割符化・復元を意識させない。また、専用画面から、ファイルの参照・更新を行うことで、終了時に自動的に割符化を行う。

- バックアップ・リストア機能/ロギング機能
  万が一の備えとして、専用 USB フラッシュメモリ内の割符ファイルのバックアップおよび、リストアが可能である。また、ユーザが専用画面で行った操作をログ情報として出力する。

2.3.2 (k,n) しきい値法

以降では、本研究で利用した秘密分散共有法の一つである“ (k,n) しきい値法 ”について述べる。

この秘密分散共有法では、秘密情報を n 人の管理者に分散させておくと、k 人未満の管理者では秘密情報は分からないが、秘密情報を紛失したときに k 人以上が集まると秘密を復元できるということが可能である。

以下では (k,n) しきい値法の概要を説明する。

(k,n) しきい値法のモデル

秘密分散共有法は、秘密情報 s の保有者 (ディーラ) と、複数の分散管理者の間で行われる。以降、ディーラを D、n 人の分散管理者を $P_1,\cdots,P_n$ で表す。秘密分散共有法は以下の図 2.2 のように分散段階と再構成段階によって構成される。

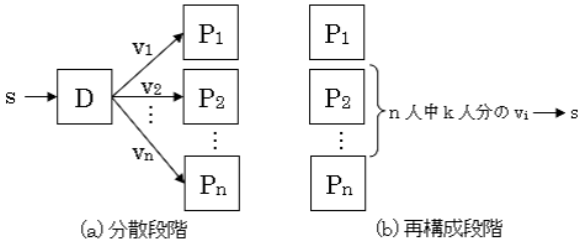


図 2.2: 秘密分散共有法

- **分散段階** 図 (a) のように、D は秘密情報 s から、n 個のシェアと呼ばれる $v_1,\cdots,v_n$ を計算し、v_i を P_i に与える。
- **再構成段階** 図 (b) のように、n 人の分散管理者のうち何人かが集まり、シェアから s を求める。

(k,n) しきい値法では、$v_1,\cdots,v_n$ は次の条件を満たすように作成されなければならない。

・$v_1,\cdots,v_n$ のうち、任意の k 個から元の秘密情報 s を復元できる。
・どの $k-1$ 個を集めても、s について何もわからない。

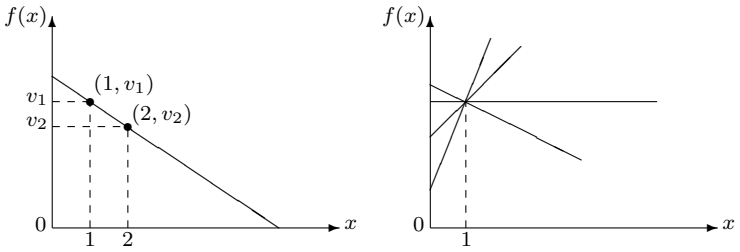
(k,n) しきい値法の例

(k,n) しきい値法の実現方法として、多項式補間を利用した Shamir(シャミア) の方法がよく知られている。以下に例を示す。

例 $(2,n)$ しきい値秘密分散共有法は、次のようになる。

- **分散段階** 秘密 s に対し D は a_1 をランダムに選び、$f(x) = s + a_1x$ とおく。D は次に、$v_i = f(i)$ を計算し、シェア v_i を P_i に与える。ただし、$1 \leqq i \leqq n$ である。

- **再構成段階** P_1, P_2 が集まったと仮定すると、図 2.3(a) に示すように 2 点 $(1, v_1), (2, v_2)$ を通る直線 $y = f(x)$ は一意に決まる。これより、$s = f(0)$ が求まる。このように、任意の 2 人が集まると s がわかる。しかし、P_i 一人だけでは、図 (b) に示すように直線は決まらないので、s について何もわからない。



(a)2 点から直線が求まる (b)1 点では直線は定まらない

図 2.3: 秘密分散共有法

2.3.3 実現方法

(k, n) しきい値法の一般的な実現方法を以下に記述する。(p が素数のとき、mod$p$ の世界を $GF(p)$ というのであった)

1. **分散段階**

   (a) D は、$p >\max(s, n)$ となる素数 p を選ぶ。

   (b) D は、$f(0) = s$ となる $GF(p)$ 上の高々$k-1$ 次の多項式
   $$f(x) = s + a_1 x + \cdots + a_{k-1}x^{k-1} \bmod p \qquad (1)$$
   をランダムに選ぶ。

   (c) D は、$v_i = f(i)$ を計算し、シェア v_i を P_i に与える。ただし、$1 \leqq i \leqq n$ である。

2. **再構成段階**

   k 人の分散管理者 $P_{i_1}, \cdots, P_{i_k}$ が集まったとき
   $$v_{i_j} = f(i_j) \bmod p \qquad (2)$$
   が $j = 1, \cdots, k$ で満たされるような高々$k-1$ 次の多項式 $f(x)$ を考える。実際、任意の $k-1$ 次以下の多項式 $f(x)$ は、$(i, f(i))$ の組が k 個あれば復元できる。具体的には、次の Lagrange(ラグランジェ) の補間公式が知られている。
   $$f(x) = \lambda_1(x)f(i_1) + \cdots + \lambda_k(x)f(i_k) \bmod p \qquad (3)$$
   ただし
   $$\lambda_j(x) = \frac{(x-i_i)\cdots\overset{j}{\vee}\cdots(x-i_k)}{(i_j-i_i)\cdots\overset{j}{\vee}\cdots(i_j-i_k)} \bmod p \qquad (4)$$
   式 (3) が成り立つことは、$x = i_1, \cdots, i_k$ を両辺に代入することにより、容易に確かめられる。式 (3) により、次式が得られる。
   $$s = f(0) = \lambda_1(0)f(i_1) + \cdots + \lambda_k(0)f(i_k) \bmod p \qquad (5)$$
   これより、任意の k 人が集まると、s を復元できることがわかる。

第3章　秘密分散システムの概要

本章では、第2章で述べてきた内容をもとに、FeliCa カードへ (k, n) しきい値秘密分散共有法を構築させる。

この提案システムによって、認証に必要とされる鍵情報のような秘密情報の管理を一か所（一部）から複数の互いに異なった情報へと分散させる。このシステムには、分散された情報からは、任意に設定された数の情報からでないと、元の秘密情報を復元することができず、設定数未満での情報からでは、秘密情報を何も知ることができない、という特徴を持っている。この秘密分散共有法の技術は“ 割符 ”の考え方を利用したものである。

このシステムを導入することによって、認証処理におけるセキュリティの安全性を向上させるだけでなく、秘密情報の紛失・漏えい、単独管理者による悪用を防ぎ、バックアップ効果を得られる。また、非接触型 IC カードの普及によるスキミング対策としても利用できる。さらに非接触技術を利用したスムーズな処理が期待できる。

3.1 システム構築の準備と手順

3.1.1 開発環境

本システムは、

- FeliCa チップを搭載した RFID
- FeliCa リーダ/ライタ
- リーダ/ライタを制御、
  および分散・再構成の計算行う上位機器

から構成される。

上位機器として、Windows を搭載したパソコンによるアプリケーションから Windows フォームを利用して、秘密情報の入力や計算などを行う。システムの構築は以下の環境で行った。

表 3.1: 開発環境

OS	Microsoft Windows XP Professional Service Pack 2
開発言語	Basic (Microsoft Visual Basic 2005 Express Edition) [16]
リーダ/ライタ	株式会社デンソーウェーブ製 PR-400UDM
カード	株式会社デンソーウェーブ発行 サンプルカード

なお、リーダ/ライタの制御はデンソーウェーブ独自のライブラリ [5] を用いることによって、プログラム中では関数として FeliCa のコマンドを利用することができる。

3.1.2 秘密分散システムの流れ

Windows フォームを利用して、以下のように分散段階（暗号）と再構成段階（復号）とに分けて処理が進行する。

分散段階（暗号）

1. 秘密情報 s の入力フォームを表示
2. テキスト入力 (半角英数字) で s を入力
3. カードに分散させる数値 n の入力フォームを表示
4. n を半角数字で入力
5. 再構成可能な数値 k の入力フォームを表示
6. k を半角数字で入力
7. 各カード $P_1, \cdots, P_n$ に書き込むデータ $v_1, \cdots, v_n$ を計算
8. 各カード P_i をリーダ/ライタにかざす確認をするフォームを表示
9. カードに v_i を書き込み
10. 8 から 9 を繰り返し（ただし $1 \leqq i \leqq n$）
11. 秘密情報の分散処理が完了

再構成段階（復号）

1. 再構成に利用するカード P_{i_j} をリーダ/ライタにかざす確認をするフォームを表示
2. カードから v_{i_j} を読み取り
3. 集めたカードの読み取りが完了するまで 1 から 2 を繰り返し（ただし $1 \leqq j \leqq k$）
4. 読み取った各 v_{i_j} から秘密情報 s を計算
5. カードの枚数・データの内容が正しければ s の復元は完了

3.2 システムのモデル

構築したシステムのモデルを分散段階、再構成段階の処理別にフローチャートで示した。分散段階の処理は図 3.2、再構成段階の処理は図 3.3 である。

今回提案するシステムでのカードに書き込むデータは、研究のしやすさを考えて以下のような形式とする。[18]

- 1 ブロック（16 バイト）ごとに、偶数ブロックを仮数部データ、奇数ブロックを指数部データとして保存する。
- 計算された結果である整数の下位 1 バイトごとに、偶数ブロックへ順次保存し、その指数は奇数ブロックへ保存する。
- 偶数ブロックの先頭バイト（0 番目）にはカード番号 i （$1 \leqq i \leqq n$）を格納し、
  奇数ブロックの先頭バイト（0 番目）には、チェック用に $(\mathrm{FF})_{16}$ を格納する。

以下の図 3.1 に、10 進整数「1234567」を保存する場合を例として示す。
（表示は 16 進数であるため $(1234567)_{10} \rightarrow (12\mathrm{D}687)_{16}$ である）
ここで、カード番号 $i = 1$、ブロック 0、1 に書き込むこととする。

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
ブロック 0	01	87	D6	12	00	00	00	00	00	00	00	00	00	00	00	00
ブロック 1	FF	01	02	03	00	00	00	00	00	00	00	00	00	00	00	00

図 3.1: 10 進整数「1234567」の保存形式

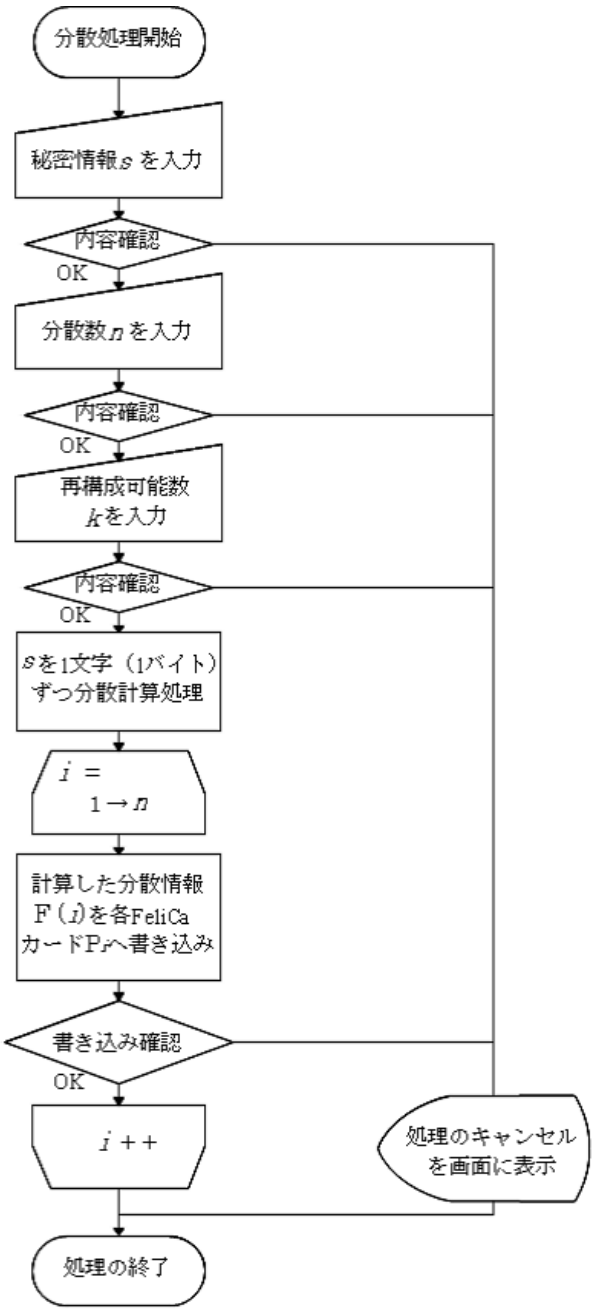


図 3.2: 分散段階の処理図

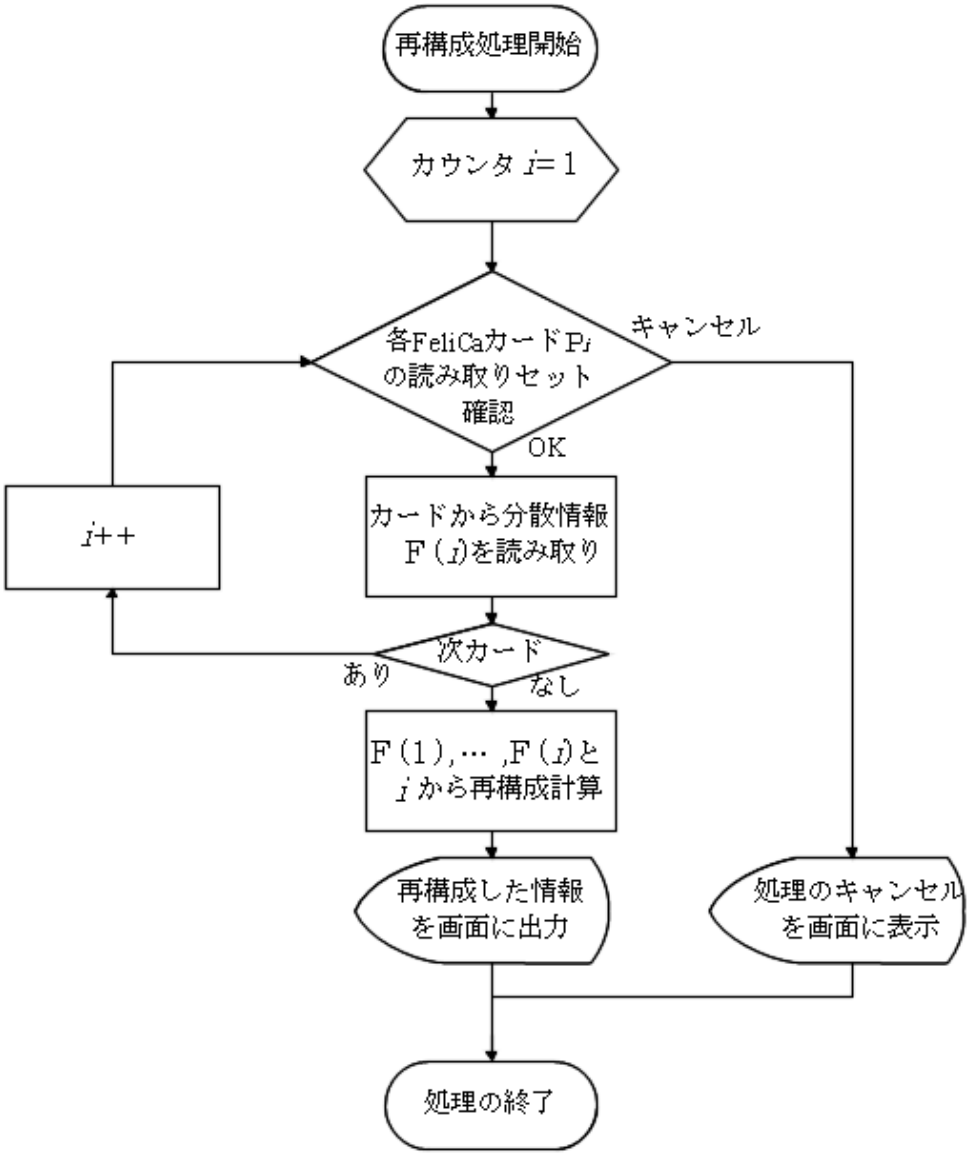


図 3.3: 再構成段階の処理図

3.3 システムの実行

提案したシステムの現段階における構築、および動作内容を次に示す。

3.3.1 実行テスト

例 以下の条件から実行テストを行った。

表 3.2: テスト内容

秘密情報 s	“ Hello!! ”
分散数 n	3
再構成可能数 k	2

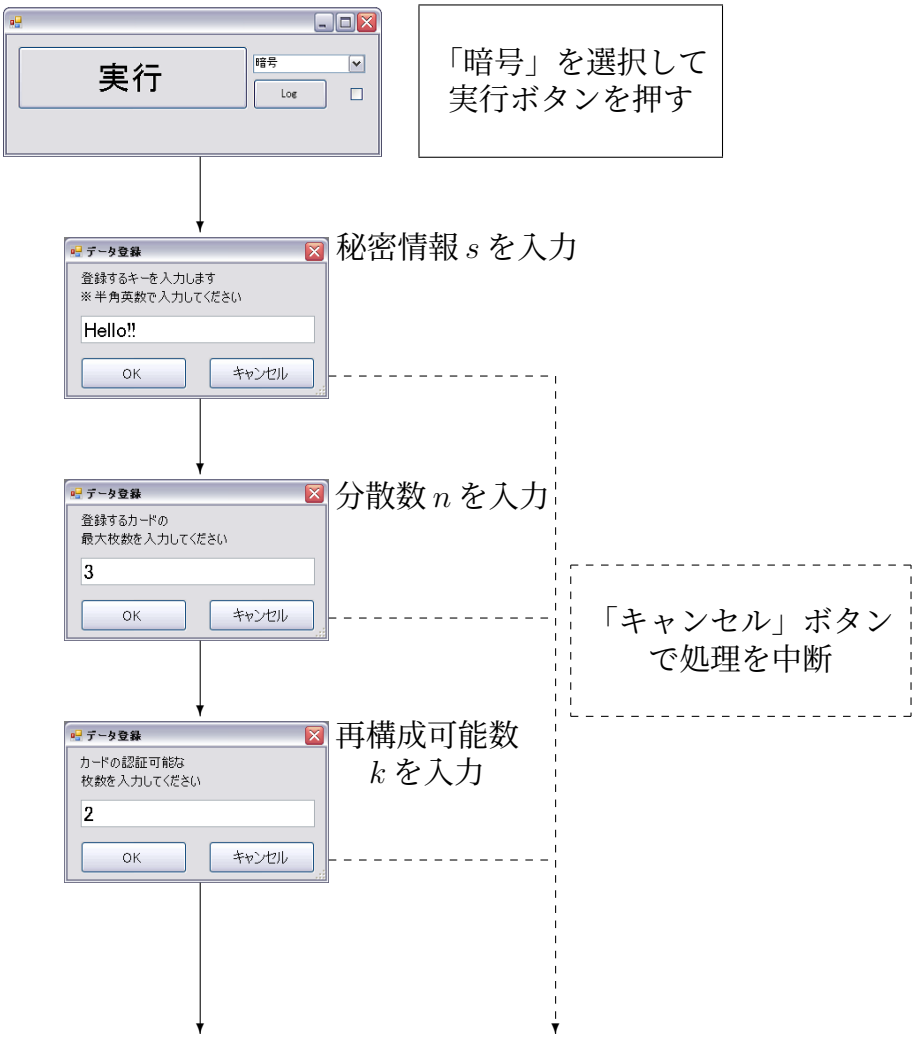
なお、秘密情報 s はバイナリデータに変換すると、

“ Hello!! ”→ 0x(48 65 6C 6F 21 21)

となる。これらを 1 文字ずつ計算して、分散処理を行う。

3.3.2 テスト結果

上記の例より各カードに分散情報を書き込み（暗号化）を行い、再構成（復号化）を行った結果、各カードすべての組み合わせにて秘密情報 s である“ Hello!! ”のデータを取得することができた。これらの実行結果を次の GUI 画面と共に次に示す。
（なお、実行内容は分散段階を暗号、再構成段階を復号とし、それぞれ図 3.4 & 3.5 および図 3.6 として区別して図に示している。）
また、カードに書き込まれたデータをそれぞれブロックごとに１６進数表示で図 3.7 に示す。



次ページへ

図 3.4: 分散処理（暗号）の実行テスト (1)

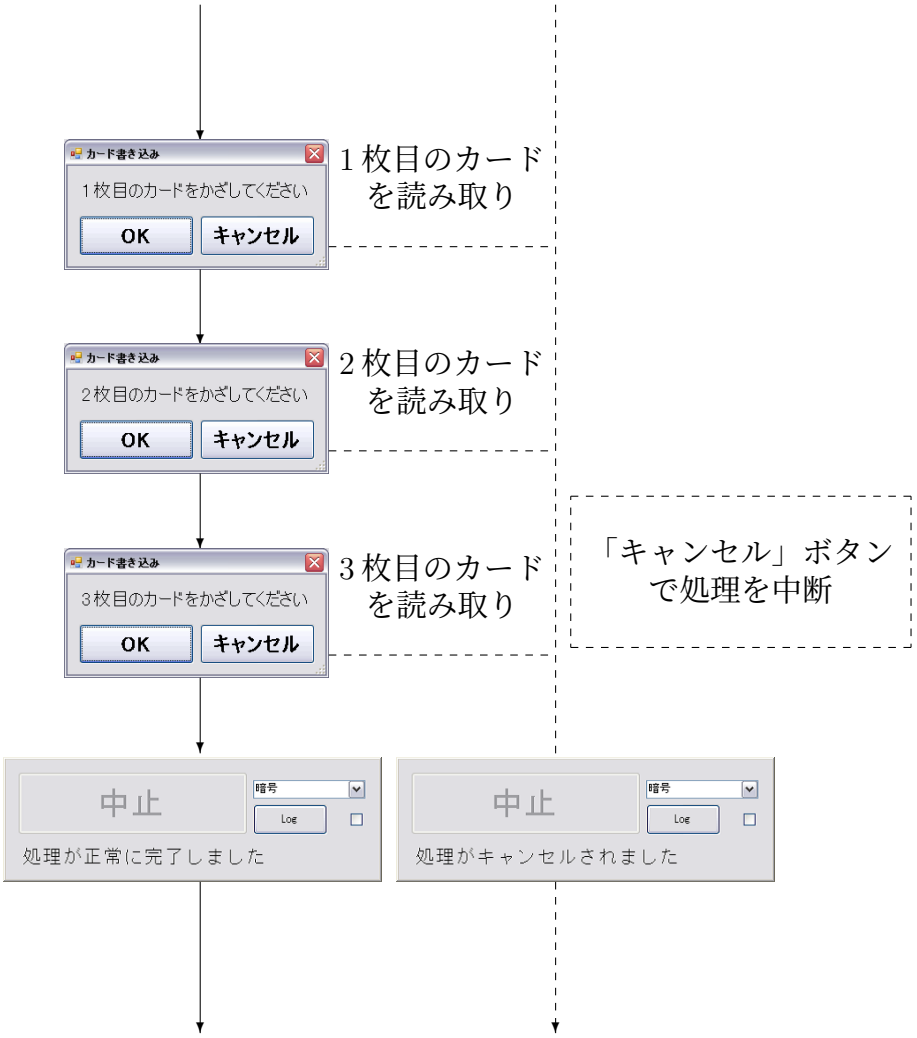


図 3.5: 分散処理（暗号）の実行テスト (2)

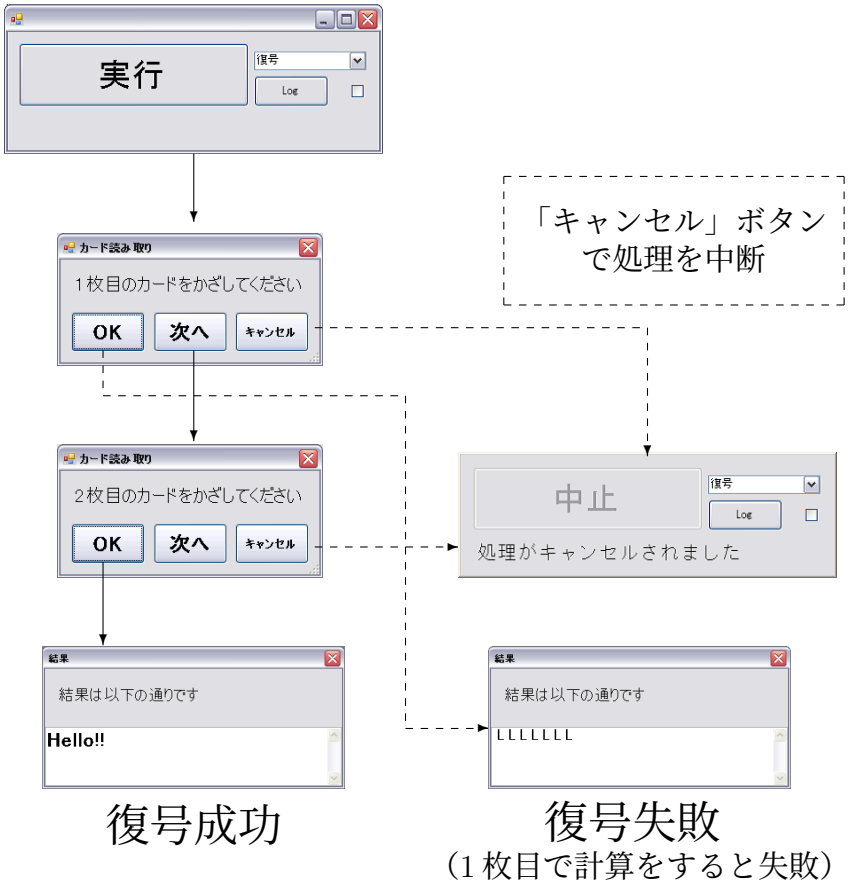


図 3.6: 再構成処理（復号）の実行テスト

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
ブロック 0	01	27	01	44	01	4B	01	4B	01	4E	01	00	01	00	01	00
ブロック 1	FF	01	02	01	02	01	02	01	02	01	02	01	02	01	02	00

(a) カード 1 のデータ

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
ブロック 0	02	06	02	23	02	2A	02	2A	02	2D	02	DF	01	DF	01	00
ブロック 1	FF	01	02	01	02	01	02	01	02	01	02	01	02	01	02	00

(b) カード 2 のデータ

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
ブロック 0	03	E5	02	02	03	09	03	09	03	0C	03	BE	02	BE	02	00
ブロック 1	FF	01	02	01	02	01	02	01	02	01	02	01	02	01	02	00

(c) カード 3 のデータ

図 3.7: 各カードから読み取ったデータ

前述の図 3.7 より、3.2 の形式を参考にして秘密情報 s の各文字の分散情報を 10 進整数値に変換すると、以下の表 3.3 に示すとおりとなった。

表 3.3: 各カードの文字毎における分散結果

文字	分散値
1（H）	295
2（e）	324
3（l）	331
4（l）	331
5（o）	334
6（!）	256
7（!）	256

(a) カード 1

文字	分散値
1（H）	518
2（e）	547
3（l）	554
4（l）	554
5（o）	557
6（!）	479
7（!）	479

(b) カード 2

文字	分散値
1（H）	741
2（e）	770
3（l）	777
4（l）	777
5（o）	780
6（!）	702
7（!）	702

(c) カード 3

これらを 2.3.2 での再構成段階による式 (5) より、各秘密情報 s を復元すると、もとの s=“ Hello!! ”と同じ数値を得ることができたため、システムの動作を確認することができた。

第4章　評価

今回提案した秘密分散システムでの実行結果から得られた評価には以下のような内容が得られた。

- (k, n) しきい値秘密分散共有法を利用することにより、任意に決定した数よりも少ない情報では、秘密情報を得ることができなくなり、一人（一つ）の管理状態に比べてセキュリティ性の向上がみられると考えられる。
- 非接触によるデータの読み書きを行うため、処理がスムーズになり、システム運用などの面で利点がある。
- 本論文で述べたテスト運用では秘密情報 s に“ Hello!! ”の 7 文字（7 バイト）を用いたが、研究の中では、“ This is the test of Secret Sharing Schemes ”の 42 文字（42 バイト）の分散が可能であった。再構成可能数（認証可能数）によって、計算量が変化するため正確な分散可能文字数を求めることができなかった。
- 再構成可能数（認証可能数）が大きくなるにつれて、分散段階における計算値が大きくなり、１つの FeliCa チップに対するデータの量が大きくなってしまう。したがって、現在の FeliCa チップのメモリ容量は数 k バイトであるため、大容量のデータ保存に向いていない可能性がある。
- 今回のカードに保存するデータ形式は研究のしやすさを考え、簡易的な形式としたため保存量が少なくなってしまったが、分散段階における、計算値の保存に関するフォーマットを変えることによって現段階以上に保存できるデータ量が増えるのではないかと考えられる。
- FeliCa チップ搭載製品の利用に際して、サービスブロックやメモリとしての使用で自主利用をするためのフォーマット（発行）が必要となってくるため、その手続きなどが必要となってくる。

次の表 4.1 には、
テキストデータ“ A ”(16 進表記で 65) 1 文字を秘密情報 s として分散した場合の、
最大分散数 n を 10 までとしたとき、
再構成可能数 k を $2 \leqq k \leqq 10$ を条件として、各カード P_x に分散（書き込み）させる値、またそのバイト数を計算したものである。

表のようにカード (n) が増加および、カード番号が大きいもの程、書き込みに要するデータ量が必要となる。また同様に、再構成可能数 k が増加するほどデータ量が 10 倍されていることが分かり、網掛けされている部分に該当する条件下では、コンピュータの整数計算が桁あふれを起こし、処理が停止してしまうという結果が出てしまう。これらを改善するために更に分散・再構成での計算、カードへの書き込みのフォーマットの変更などが課題となった。

表 4.1: 分散条件の違いによる変換後データ（容量）の比較

( ) 内はバイト数

	$f(x)$								
カード P_x	$k=2$	$k=3$	$k=4$	$k=5$	$k=6$	$k=7$	$k=8$	$k=9$	$k=10$
1	188 (1)	311 (2)	434 (2)	557 (2)	680 (2)	803 (2)	926 (2)	1049 (2)	1172
2	311 (2)	803 (2)	1787 (2)	3755 (2)	7691 (2)	15563 (2)	31307 (2)	62795 (2)	125771
3	434 (2)	1541 (2)	4862 (2)	14825 (2)	44714 (2)	134381 (3)	403382 (3)	1210385 (3)	3631394
4	557 (2)	2525 (2)	10397 (2)	41885 (2)	167837 (3)	671645 (3)	2686877 (3)	10747805 (3)	42991517
5	680 (2)	3755 (2)	19130 (2)	96005 (3)	480380 (3)	2402255 (3)	12011630 (3)	60058505 (4)	300292880
6	803 (2)	5231 (2)	31799 (2)	191207 (3)	1147655 (3)	6886343 (3)	41318471 (4)	247911239 (4)	1487467847
7	926 (2)	6953 (2)	49142 (2)	344465 (3)	2411726 (3)	16882553 (4)	118178342 (4)	827248865 (4)	5790742526
8	1049 (2)	8921 (2)	71897 (3)	575705 (3)	4606169 (3)	36849881 (4)	294799577 (4)	2358397145	18867177689
9	1172 (2)	11135 (2)	100802 (3)	907805 (3)	8170832 (3)	73538075 (4)	661843262 (4)	5956589945	53609310092
10	1295 (2)	13595 (2)	136595 (3)	1366595 (3)	13666595 (3)	136666595 (4)	1366666595 (4)	13666666595	136666666595

第5章　結論

現在、秘密分散共有法を適用した製品（電子割符 [12]）として USB メモリを利用したものがある。これは、コンピュータに搭載されているハードディスク（以下、HDD）に保存する秘密データを分散させ、その一部を USB メモリに格納することによって次回からの利用では USB メモリをコンピュータに接続して認証するという方法である。この製品の場合は、データの一部が認証を行う本体（コンピュータ）の HDD 内に保存されており、データの漏えいなどのセキュリティ性に少々不安がある。また、USB 接続であるため接触点が必要となってしまい、スムーズな認証が行えなくなるほか、現在最も汎用性があるインターフェースであるため、こちらにもセキュリティ性の面でも不安点がある。したがって、本研究での FeliCa を電子割符として、秘密分散共有法を適用したシステムを用いることにより、スムーズかつ安全な認証・データ暗号が期待できるのではないかと考えられる。また、前章にてメモリ容量が少ないことを述べたが、技術向上によりこの問題も徐々に解決されるものと考えられる。これには、現在の秘密分散法では処理に時間がかかってしまうことが問題とされていたが、数値演算を行う処理をコンピュータが効率よく処理できる論理演算のみを用いることによって高速化するという研究成果が株式会社 KDDI 研究所 [13] によって発表された。このアルゴリズムを用いることによって処理の高速化だけでなく、データ量の縮小化も実現できるのではないかと考えられる。FeliCa チップ（カード）の発行に関しても、セキュリティカードの利用としてこのシステムが多く利用されてくるのであれば、専用のフォーマットなどがされてくるのではないかと考えられる。

謝辞

本研究を行うにあたり、終始熱心に様々な面で数多くの有益なご助言・ご指導をしていただいた木下宏揚教授に心から感謝いたします。また、ご多忙の折大学に足を運びご指導いただいたネッツエスアイ東洋株式会社の森住哲也氏、ならびに研究活動に様々なご助力をいただきました鈴木一弘氏に深く感謝いたします。さらに、公私にわたり良き研究生活を送らせていただいた木下研究室の方々に感謝いたします。

参考文献

[1] SonyJapanFeliCa
＜ http://www.sony.co.jp/Products/felica/＞

[2] 『現代暗号の基礎数理』黒澤馨/尾形わかは
（コロナ社,2004）

[3] Microsoft: 流通サービスサイト RFID 入門
＜ http://www.microsoft.com/japan/business/rfid/about/default.mspx ＞

[4] FeliCa Networks （フェリカネットワークス株式会社）
＜ http://www.felicanetworks.co.jp/index.html ＞

[5] 機密による外部秘書類
デンソーウェーブより リーダ/ライタおよびソフトウェア付属使用説明書
「AID リファレンスマニュアル」
「PR-400UDM UsersManual」
「PR-CommandManual」
「PR 用開発支援ライブラリ 300-500F」
「PR-301/PR-400/PR-500FeliCa カード制御コマンドマニュアル」
「PR-301/PR-400/通信セキュリティマニュアル」
「サンプルカードフォーマット仕様表」

[6] 土居絵美 指導教員 清水明宏
高知工科大学 フロンティア工学コース 学士学位論文
携帯電話端末を用いた FeliCa 統合認証システム
＜ http://www.kochi-tech.ac.jp/library/ron/2006/2006info/
full/1070425.pdf ＞

[7] 寺尾良 指導教員 清水明宏
高知工科大学 情報システム工学科 学士学位論文
RFID システムへのワンタイムパスワード認証方式の適用
＜ http://www.kochi-tech.ac.jp/library/ron/2004/2004info/1050344.pdf ＞

[8] 井山幸大 指導教員 野口健一郎
FeliCa を利用したインテリジェント学生証の実装実験
＜ http://www.nol.info.kanagawa-u.ac.jp/research/2006/iyama.pdf ＞

[9] 非接触 IC に最適化された「FeliCa」の正体－@ IT
＜ http://www.atmarkit.co.jp/frfid/special/felica/felica01.html ＞

[10] トッパン・フォームズ株式会社 FeliCa
＜ http://www.rdsc.jp/felica/index.html ＞

[11] 注目の情報管理方式「しきい値秘密分散法」－@ IT
＜ http://www.atmarkit.co.jp/fsecurity/special/53tsss/tsss.html ＞

[12] HITACHI 電子割符シリーズ　株式会社 日立製作所
＜ http://www.hitachi.co.jp/Prod/comp/warifu/index.html ＞

[13] 超高速秘密分散方式の開発　株式会社 KDDI 研究所
＜ http://www.kddilabs.jp/press/detail_99.html ＞

[14] Suica ホームページ
＜ http://www.jreast.co.jp/suica ＞

[15] PASMO ホームページ
＜ http://www.pasmo.co.jp/＞

[16] Microsoft Visual Studio ホームページ
＜ http://www.microsoft.com/japan/msdn/vstudio ＞

[17] 東光電気株式会社 FeliCa
＜ http://www.tokodenki.co.jp/solution/felica/index.html ＞

[18] 『プログラミングに活かす データ構造とアルゴリズムの基礎知識』
今泉貴史（ASCII,2004）

付録

システム構築にあたって

本論文における秘密分散共有法を適用するシステムの構築には、株式会社デンソーウェーブ製のリーダ/ライタ（PR-400UDM）を使用するが、この PR-400UDM をプログラム中で使用するための開発環境ライブラリ（AID）をインストールする必要がある。インストール、および使用方法については同社配布のマニュアル・仕様書を参照する。本章では、実際に研究で行った方法等をマニュアル等から抜粋して記述すると共に、使用する関数のパラメータ記述方法など、システム構築にあたっての手順等を記載する。

開発支援ライブラリのインストール

以下は、FeliCa チップ用ライブラリをインストールする手順である。
※「AID リファレンスマニュアル」を参照

1. Windows がインストールされているドライブ（C）に“ AID ”フォルダを作成する。
2. 作成したフォルダ内に“ PR 用開発支援ライブラリ 300/500F for Windows (AD-300PR-1F Ver 2.1) ”とラベル表記された CD-ROM から、以下のファイルをコピーする。
   - AIDDEV.DLL
   - AIDDEVCOM.DLL
   - AIDICC.DLL
   - AIDICCPROFEL2.dll
   - AIDICCTRNCF.dll
   - AIDIFD.DLL
   - AIDIFDPR100.DLL
   - AIDIFDPR300.dll
   - AIDAPIFEL2.dll

3. メモ帳などで次の内容を入力したテキストファイルを作成する。
   ただし、ファイル名は「install.bat」として保存する。

```
REGSVR32␣/s␣C:\AID\AIDDEV.DLL
REGSVR32␣/s␣C:\AID\AIDDEVCOM.DLL
REGSVR32␣/s␣C:\AID\AIDICC.DLL
REGSVR32␣/s␣C:\AID\AIDICCPROFEL2.dll
REGSVR32␣/s␣C:\AID\AIDICCTRNCF.dll
REGSVR32␣/s␣C:\AID\AIDIFD.DLL
REGSVR32␣/s␣C:\AID\AIDIFDPR100.DLL
REGSVR32␣/s␣C:\AID\AIDIFDPR300.dll
REGSVR32␣/s␣C:\AID\AIDAPIFEL2.dll
```

4. 作成した「install.bat」をダブルクリックすると AID がインストールが完了する。

PR-400UDM ドライバのインストール

PR-400UDM の USB ドライバをインストールする。ドライバは

＜ http://www.denso-wave.com/ids/download/driver.html ＞

からダウンロードする。

インストール手順はWindowsのバージョンによって異なるため、「PR-400UDM 取扱説明書」を参照して行う。

テスト用サンプル FeliCa カード

本研究に用いるテスト用カードは、デンソーウェーブ発行のサンプルカードフォーマットされたカードを利用する。

カードのシステムコード・サービスコード等については、デンソーウェーブ発行の「サンプルカードフォーマット仕様表」を参照。

サンプルプログラム

サンプルプログラムの入手

FeliCa カード読み取りの基本となるサンプルプログラムをダウンロードする。入手先は、

＜ http://www.denso-wave.com/ja/adcd/support/index.html ＞

である。サンプルプログラムは、Microsoft Visual Basic 6.0 および Microsoft Visual C++ 6.0 用が用意されている。本研究では Microsoft Visual Basic 6.0 用のファイルを利用した。

サンプルプログラムの実行

サンプルプログラム“ SampleFel2 ”のプロジェクトファイルを開き、デバッグを実行すると、以下の図 5.1 の画面が表示される。

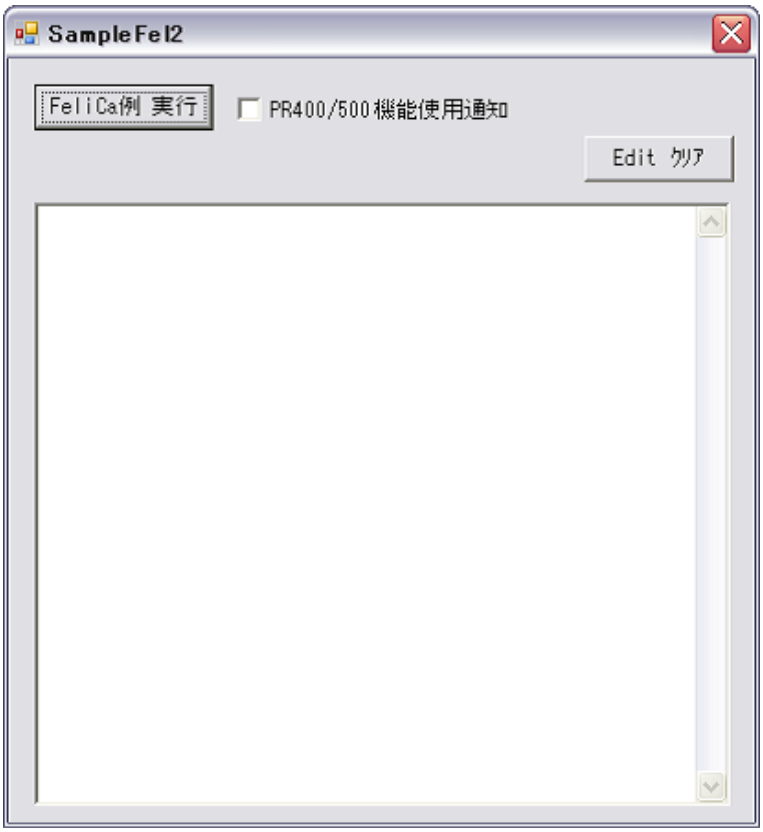


図 5.1: サンプルプログラムの GUI

プログラム中に以下の条件を記述して、サンプルカードに書き込まれたデータを読み取る処理を実行する。

表 5.1: サンプルカードの読み取り条件

システムコード	0x0000
サービス数	1
サービスコード	0x1009（認証なしサービス）
ブロック数	1
ブロックリスト	0x8000

これらの設定は SetStart_felmode1_Ex() での vvalue および、Fel_Read_Ex() での vsList、vbList にて変更を行うことによって、FeliCa カードのサービス・ブロックを読み込むことができる。実行結果の GUI は以下の図 5.2 のように表示される。網掛け部分が読み取ったデータである。

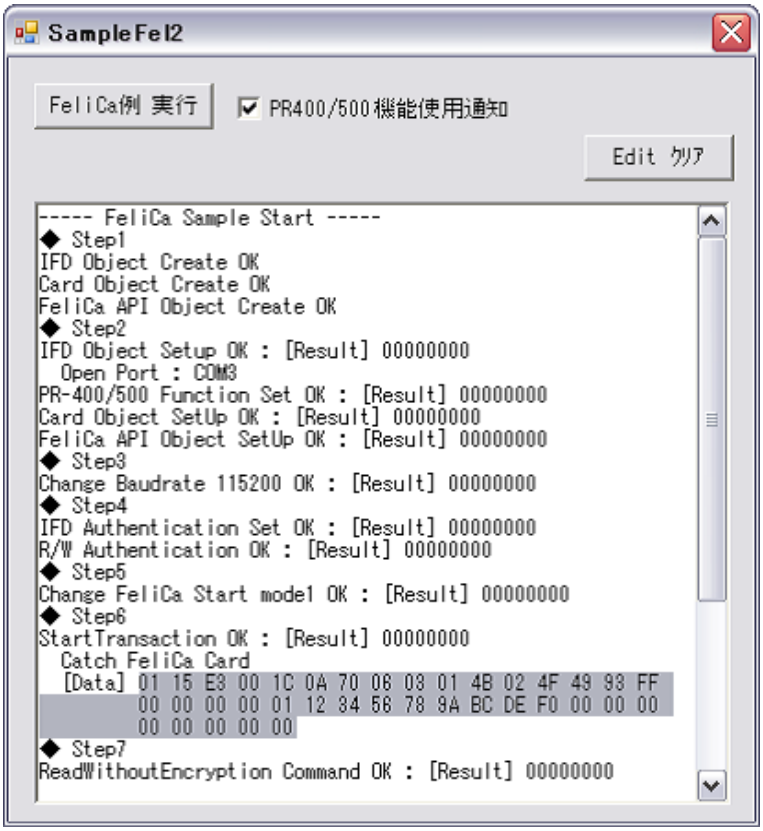


図 5.2: サンプルプログラムの実行結果

なお、プログラムの内容については以降のページに記載する。

サンプルプログラム

```
Option Strict Off
Option Explicit On
Imports VB = Microsoft.VisualBasic
Friend Class FormMain
Inherits System.Windows.Forms.Form


'// 各オブジェクトの変数を宣言
Public ifd As Object '// IFD オブジェクト
Public icc As Object '// Card オブジェクト
Public fel As Object '// FeliCa オブジェクト


'//=============================================================
'// AID を使った一連の手順例
'//=============================================================
Private Sub OnFeliCa_Exe_Click(ByVal eventSender _
                As System.Object, ByVal eventArgs _
                  As System.EventArgs) Handles OnFeliCa_Exe.Click

Dim result As Integer


Call upDateEdit_1("----- FeliCa Sample Start -----")
Do

'//-------------------------------------------------------
'◆ Step1：IFD/カード/API オブジェクトの生成

Call upDateEdit_1("◆ Step1")
result = CreateObject_Ex()
If result <> NORMALEND Then
Exit Do
End If


'//-------------------------------------------------------
```

```
'◆ Step2：IFD/カード/API オブジェクトに対する SetUp

Call upDateEdit_1("◆ Step2")
result = ObjectSetUp_Ex()
If result <> NORMALEND Then
Exit Do
End If


'//-------------------------------------------------------
'◆ Step3：ホストとリーダ/ライタ間の通信速度を
'   115200bps へ変更（デフォルトは 9600bps です）
'   変更不要な場合、この処理は削除可能です

Call upDateEdit_1("◆ Step3")
result = ChangeSpeed_Ex(DEF_115200)
If result <> NORMALEND Then
Exit Do
End If


'//-------------------------------------------------------
'◆ Step4：ホストとリーダ/ライタ間の相互認証

Call upDateEdit_1("◆ Step4")
result = RW_Authentication_Ex()
If result <> NORMALEND Then
Exit Do
End If


'//-------------------------------------------------------
'◆ Step5：FeliCa カードに対する StartTransaction で実行
'   する内容をモード 1 へ変更
'   デフォルト設定を使用する場合、この処理は不要です

Call upDateEdit_1("◆ Step5")
result = SetStart_felmode1_Ex()
If result <> NORMALEND Then
Exit Do
```

```
End If


'//-------------------------------------------------------
'◆ Step6：StartTransaction を実行しカードを捕捉

Call upDateEdit_1("◆ Step6")
result = CatchCard_Ex()
If result <> NORMALEND Then
Exit Do
End If


'//-------------------------------------------------------
'◆ Step7：ReadWithoutEncryption を実行しカード内のデータ読み出し

Call upDateEdit_1("◆ Step7")
result = Fel_Read_Ex()
If result <> NORMALEND Then
     Exit Do
End If


'//-------------------------------------------------------
'◆ Step8：EndTransaction を実行しカードを開放

Call upDateEdit_1("◆ Step8")
result = ReleaseCard_Ex()
If result <> NORMALEND Then
     Exit Do
End If


'//-------------------------------------------------------
'◆ Step9：ホストとリーダ/ライタ間の通信速度を 9600bps へ変更
'    変更不要な場合、この処理は削除可能です

Call upDateEdit_1("◆ Step9")
result = ChangeSpeed_Ex(DEF_9600)
If result <> NORMALEND Then
```

```
     Exit Do
End If


Loop While (0)


'//-------------------------------------------------------------
'◆ Step10：IFD/カード/API オブジェクトの開放

Call upDateEdit_1("◆ Step10")
Call ReleaseAid_Ex()


Call upDateEdit_1("----- FeliCa Sample End -----" & vbCrLf)

End Sub

'//=============================================================
'// IFD/カード/API オブジェクトの生成例です
'//=============================================================
Private Function CreateObject_Ex() As Short

On Error Resume Next

'// オブジェクトが既に存在する場合は開放
If ((ifd Is Nothing) = False) Or ((icc Is Nothing) _
                  = False) Or ((fel Is Nothing) = False) Then
Call ReleaseAid_Ex()
End If


'//-------------------------------------------------------------
'// IFD オブジェクトの生成
ifd = CreateObject("IFD.RW")
If (ifd Is Nothing) <> False Then
Call upDateEdit_1("IFD Object Create Error")
CreateObject_Ex = 1
Exit Function
End If
```

```
Call upDateEdit_1("IFD Object Create OK")


'//-------------------------------------------------------------
'// カードオブジェクトの生成
icc = CreateObject("ICC.Card")
If (icc Is Nothing) <> False Then
Call upDateEdit_1("Card Object Create Error")
CreateObject_Ex = 1
Exit Function
End If
Call upDateEdit_1("Card Object Create OK")


'//-------------------------------------------------------------
'// FeliCa API オブジェクトの生成
fel = CreateObject("FeliCa2.API")
If (fel Is Nothing) <> False Then
Call upDateEdit_1("FeliCa API Object Create Error")
CreateObject_Ex = 1
Exit Function
End If
Call upDateEdit_1("FeliCa API Object Create OK")

CreateObject_Ex = NORMALEND

End Function

'//=============================================================
'// IFD/カード/API オブジェクトに対する SetUp 例です
'// (PR-400/500 用機能使用通知を含みます)
'//=============================================================
Private Function ObjectSetUp_Ex() As Integer

Dim result As Integer
Dim portName As String
Dim i As Short


'//-------------------------------------------------------------
```

```
'◆ IFD オブジェクトの SetUp
'   COM1～COM9 まで順番に接続していきます
'(COM10 以上接続時は、変更が必要です)

For i = 1 To 9
portName = "COM" & CStr(i)
result = ifd.SetUp("PR300", portName)
If result = NORMALEND Then
Exit For
End If
Next i
If result <> NORMALEND Then
Call upDateEdit_2("IFD Object Setup Error : ", result)
ObjectSetUp_Ex = result
Exit Function
End If
Call upDateEdit_2("IFD Object Setup OK : ", result)
Call upDateEdit_1("  Open Port : " & portName)


'//-------------------------------------------------------------
'◆ PR-400/500 用機能使用通知
'   (チェックボックスのチェック有無により実行可否を判定)
'   本例では、この処理を実行しなくても動作可能です
'   (PR-301 使用時はを外してください)

If Check1.CheckState = 1 Then
result = SetPR400_500FuncUse_Ex()
If result <> NORMALEND Then
ObjectSetUp_Ex = result
Exit Function
End If
End If


'//-------------------------------------------------------------
'◆ カードオブジェクトの SetUp

result = icc.SetUp("CF:FEL2", ifd)
If result <> NORMALEND Then
```

```
Call upDateEdit_2("Card Object SetUp Error : ", result)
ObjectSetUp_Ex = result
Exit Function
End If
Call upDateEdit_2("Card Object SetUp OK : ", result)


'//-------------------------------------------------------------
'◆ API オブジェクトの SetUp

result = fel.SetUp(icc)
If result <> NORMALEND Then
Call upDateEdit_2("FeliCa API Object SetUp Error : ", result)
ObjectSetUp_Ex = result
Exit Function
End If
Call upDateEdit_2("FeliCa API Object SetUp OK : ", result)

ObjectSetUp_Ex = result

End Function

'//=============================================================
'// IFD/カード/API オブジェクトの開放例です
'//=============================================================
Private Sub ReleaseAid_Ex()

'// FeliCa API オブジェクトの開放
If (fel Is Nothing) = False Then
fel = Nothing '// Release API オブジェクト
End If

'// カードオブジェクトの開放
If (icc Is Nothing) = False Then
icc = Nothing '// Release カードオブジェクト
End If

'// IFD オブジェクトの開放
If (ifd Is Nothing) = False Then
ifd = Nothing '// Release IFD オブジェクト
```

```
End If

Call upDateEdit_1("Release All Aid Object OK")

End Sub

'//=============================================================
'// PR-400/500 機能使用通知例です
'//=============================================================
Private Function SetPR400_500FuncUse_Ex() As Integer

Dim result As Integer
Dim vtag As Integer
Dim vvalue(256) As Byte
Dim vinfBuff(256) As Byte
Dim vbuffSize As Integer
Dim vinfLen(1) As Integer


vtag = &H4080500 '// 制御タグ値
vvalue(0) = &H0 '// リーダ/ライタ情報書き込み命令
vvalue(1) = &H1 '// PR-400/PR-500 用機能を使用

vbuffSize = 256

result = ifd.SetControl(vtag, vvalue, vinfBuff, _
                                          vbuffSize, vinfLen)

If result <> NORMALEND Then
Call upDateEdit_2("PR-400/500 Function Set Error : ", result)
Else
Call upDateEdit_2("PR-400/500 Function Set OK : ", result)
End If

SetPR400_500FuncUse_Ex = result

End Function

'//=============================================================
'// ホストとリーダ/ライタ間の通信速度の変更例です
```

```
'//=============================================================
Private Function ChangeSpeed_Ex(ByVal SetSpeed As Integer) _
                                                    As Integer

Dim result As Integer
Dim vtag As Integer
Dim vvalue(256) As Integer
Dim vinfBuff(256) As Byte
Dim vbuffSize As Integer
Dim vinfLen(1) As Integer


vtag = &H4010007 '// 制御タグ値
vvalue(0) = SetSpeed '// 設定する通信速度：引数で指定 (bps)

vbuffSize = 256

result = ifd.SetControl(vtag, vvalue, vinfBuff, _
                                            vbuffSize, vinfLen)

If result <> NORMALEND Then
If SetSpeed = DEF_9600 Then
Call upDateEdit_2("Change Baudrate 9600 Error : ", result)
Else
Call upDateEdit_2("Change Baudrate 115200 Error : ", result)
End If
Else
If SetSpeed = DEF_9600 Then
Call upDateEdit_2("Change Baudrate 9600 OK : ", result)
Else
Call upDateEdit_2("Change Baudrate 115200 OK : ", result)
End If
End If

ChangeSpeed_Ex = result

End Function

'//=============================================================
'// ホストとリーダ/ライタ間での相互認証例です
```

```
'//============================================================
Private Function RW_Authentication_Ex() As Integer

Dim result As Integer
Dim vtag As Integer
Dim vvalue(256) As Byte
Dim vinfBuff(256) As Byte
Dim vbuffSize As Integer
Dim vinfLen(1) As Integer


'//------------------------------------------------------------
'◆ IFD オブジェクトにリーダ/ライタ側の相互認証鍵と
'   暗号モードを設定しますリーダ/ライタの相互認証鍵、
'   暗号モードが初期状態の場合の処理例です

vtag = &H4088010 '// 制御タグ値

vvalue(0) = &H3 '// 初期化設定条件

vvalue(1) = &HFF '// 現在のリーダ/ライタの相互認証鍵 1（初期値）
vvalue(2) = &HFF
vvalue(3) = &HFF
vvalue(4) = &HFF
vvalue(5) = &HFF
vvalue(6) = &HFF
vvalue(7) = &HFF
vvalue(8) = &HFF

vvalue(9) = &HFF '// 現在のリーダ/ライタの相互認証鍵 2（初期値）
vvalue(10) = &HFF
vvalue(11) = &HFF
vvalue(12) = &HFF
vvalue(13) = &HFF
vvalue(14) = &HFF
vvalue(15) = &HFF
vvalue(16) = &HFF

vvalue(17) = &HFF '// 現在のリーダ/ライタの相互認証鍵 3（初期値）
vvalue(18) = &HFF
```

```
vvalue(19) = &HFF
vvalue(20) = &HFF
vvalue(21) = &HFF
vvalue(22) = &HFF
vvalue(23) = &HFF
vvalue(24) = &HFF

vvalue(25) = &H0 '// 認証アルゴリズム：適用なし
vvalue(26) = &H0 '// 暗号通信機能：適用なし
vvalue(27) = &H0 '// 認証暗号設定情報の暗号化：適用なし

vbuffSize = 256

result = ifd.SetControl(vtag, vvalue, vinfBuff, _
                                          vbuffSize, vinfLen)

If result <> NORMALEND Then
Call upDateEdit_2("IFD Authentication Set Error : ", result)
RW_Authentication_Ex = result
Exit Function
End If
Call upDateEdit_2("IFD Authentication Set OK : ", result)


System.Array.Clear(vvalue, 0, vvalue.Length)
System.Array.Clear(vinfLen, 0, vinfLen.Length)


'//-------------------------------------------------------------
'◆ IFD オブジェクトで、リーダ/ライタと相互認証１・２を実行

vtag = &H4088003 '// 制御タグ値
vvalue(0) = 0

result = ifd.SetControl(vtag, vvalue, vinfBuff, _
                                          vbuffSize, vinfLen)

If result <> NORMALEND Then
Call upDateEdit_2("R/W Authentication Error : ", result)
Else
```

```
Call upDateEdit_2("R/W Authentication OK : ", result)
End If

RW_Authentication_Ex = result

End Function

'//=============================================================
'// FeliCa カードに対する StartTransaction で
'// 実行する内容をモード 1 へ変更する例です
'//=============================================================
Private Function SetStart_felmode1_Ex() As Integer

Dim result As Integer
Dim vtag As Integer
Dim vvalue(256) As Byte
Dim vinfBuff(256) As Byte
Dim vbuffSize As Integer
Dim vinfLen(1) As Integer


vtag = &H2200008 '// 制御タグ値

        vvalue(0) = &H1 '// 設定条件：モード 1

vvalue(1) = &H1 '// 設定数

        vvalue(2) = &HFF '// システムコード：0x0000
        vvalue(3) = &HFF '// (ビッグエンディアンで指定)

vvalue(4) = &H1 '// サービス数：0x01
        vvalue(5) = &H9 '// サービスリスト：0x1009
vvalue(6) = &H10 '// (リトルエンディアンで指定)

vvalue(7) = &H1 '// ブロック数：0x01
vvalue(8) = &H80 '// ブロックリスト：0x8000
vvalue(9) = &H0  '// (ビッグエンディアンで指定)

vbuffSize = 256
```

```
result = icc.SetControl(vtag, vvalue, vinfBuff, _
                                        vbuffSize, vinfLen)

If result <> NORMALEND Then
Call upDateEdit_2("Change FeliCa Start mode1 Error : ", result)
Else
Call upDateEdit_2("Change FeliCa Start mode1 OK : ", result)
End If

SetStart_felmode1_Ex = result

End Function

'//=============================================================
'// StartTransaction を実行しカードを捕捉する例です
'//=============================================================
Private Function CatchCard_Ex() As Integer

Dim result As Integer
Dim vmode As Integer
Dim vspeed As Integer
Dim vtype(1) As Integer
Dim vinfBuff(256) As Byte
Dim vbuffSize As Integer
Dim vinfLen(1) As Integer


vmode = 1000 '// 検出タイムアウト時間を 1000ms とします
vspeed = CARDSPEED_DEFAULT '// カードとの初期通信速度
vtype(0) = 0 '// カード検出タイプ

vbuffSize = 256

'// カードオブジェクトを使用した StartTransaction の実行
result = icc.StartTransaction(vmode, vspeed, vtype, _
                        vinfBuff, vbuffSize, vinfLen)

        If result <> NORMALEND Then
            Call upDateEdit_1(" トランザクションエラー")
            Call upDateEdit_2("StartTransaction Error : ", _
```

```
                                                    result)
            CatchCard_Ex = result
            Exit Function
        End If
Call upDateEdit_2("StartTransaction OK : ", result)


'// 捕捉したカードの処理例

'// FeliCa カードの捕捉に成功
If (vtype(0) And (Not CARDTYPE_MORECARD)) = CARDTYPE_C Then
'// FeliCa カード捕捉時の処理を記述します
Call upDateEdit_1("  Catch FeliCa Card")

'// FeliCa カード以外のカードを捕捉（通常ない）
Else
'// FeliCa 以外のカード捕捉時の処理を記述します
Call upDateEdit_1("  Catch Another Card")

End If
        Call upDateEdit_3(vinfBuff, vinfLen(0))

'// 複数カード捕捉
If (vtype(0) And CARDTYPE_MORECARD) Then
'// 他のカード捕捉時の処理を記述します
Call upDateEdit_1("  More Card Exist")

End If

CatchCard_Ex = result

End Function

'//=============================================================
'// ReadWithoutEncryption を実行しカード内のデータを読み出す例です
'//=============================================================
Private Function Fel_Read_Ex() As Integer

Dim result As Integer
Dim vsCount As Byte
```

```
Dim vsList(1) As Byte
Dim vbCount As Byte
Dim vbList(1) As Byte
Dim vrxBuff(256) As Byte
Dim vbuffSize As Integer
Dim vrxLen(1) As Integer


vsCount = &H1 '// サービス数：0x01
        vsList(0) = &H9 '// サービスリスト：0x100D
        vsList(1) = &H10 '// (リトルエンディアンで指定)

        vbCount = &H1 '// ブロック数：0x01
        vbList(0) = &H80 '// ブロックリスト：0x8000
        vbList(1) = &H0  '// (ビッグエンディアンで指定)

vbuffSize = 256

result = fel.ReadWithoutEncryption(vsCount, vsList, _
                vbCount, vbList, vrxBuff, vbuffSize, vrxLen)

If result <> NORMALEND Then
Call upDateEdit_2( _
              "ReadWithoutEncryption Command Error : ", result)
Fel_Read_Ex = result
Exit Function
End If
Call upDateEdit_2("ReadWithoutEncryption Command OK : ", result)

'// ReadWithoutEncryption コマンドにて取得したデータの解釈
'   正常データ読み出し時、ステータスフラグ 1=0x00、
' ステータスフラグ 2=0x00、ブロック数、ブロックデータが格納されます
If (vrxLen(0) >= 2) And (vrxBuff(0) = &H0) And _
                                        (vrxBuff(1) = &H0) Then
Call upDateEdit_1("  ReadWithoutEncryption Data Contents OK")
Else
Call upDateEdit_1("  ReadWithoutEncryption Data Contents Error")
End If
Call upDateEdit_3(vrxBuff, vrxLen(0))
```

```
Fel_Read_Ex = result

End Function


'//=============================================================
'// EndTransaction を実行しカードを開放する例です
'//=============================================================
Private Function ReleaseCard_Ex() As Integer

Dim result As Integer
Dim vmode As Integer


vmode = 0 '// 開放条件：モード 0

'// カードオブジェクトを使用した EndTransaction の実行
result = icc.EndTransaction(vmode)

If result <> NORMALEND Then
Call upDateEdit_2("EndTransaction Error : ", result)
Else
Call upDateEdit_2("EndTransaction OK : ", result)
End If

ReleaseCard_Ex = result

End Function

'//=============================================================
'// Edit へのコメント、result、Data を出力
'//=============================================================
Private Sub upDateEdit_1(ByVal OutMsg As String)

txtEdit.Text = txtEdit.Text & OutMsg & vbCrLf
txtEdit.SelectionStart = Len(txtEdit.Text)

System.Windows.Forms.Application.DoEvents()

End Sub
```

```
Private Sub upDateEdit_2(ByVal OutMsg As String, _
                                    ByVal result As Integer)

txtEdit.Text = txtEdit.Text & OutMsg & "[Result] " & _
              VB6.Format(Hex(result), "00000000") & vbCrLf
txtEdit.SelectionStart = Len(txtEdit.Text)

System.Windows.Forms.Application.DoEvents()

End Sub

Private Sub upDateEdit_3(ByRef data() As Byte, _
                                        ByVal dLen As Integer)
Dim tmpStr As String
Dim i As Short

tmpStr = "  [Data] "

For i = 1 To dLen
tmpStr = tmpStr & VB.Right("0" & Hex(data(i - 1)), 2) & " "

If (i Mod 16 = 0) And (i <> dLen) Then
tmpStr = tmpStr & vbCrLf & "          "
End If
Next i

txtEdit.Text = txtEdit.Text & tmpStr & vbCrLf
txtEdit.SelectionStart = Len(txtEdit.Text)

System.Windows.Forms.Application.DoEvents()

    End Sub
    Private Sub upDateEdit_4(ByVal result As Integer)

        txtEdit.Text = txtEdit.Text & "[Result]の内容=> _
            " & VB6.Format(Hex(result), "00000000") & vbCrLf
        txtEdit.SelectionStart = Len(txtEdit.Text)

        System.Windows.Forms.Application.DoEvents()
```

```
    End Sub

'//=============================================================
'// Editのクリア
'//=============================================================
Private Sub OnEditCrear_Exe_Click(ByVal eventSender _
           As System.Object, ByVal eventArgs _
            As System.EventArgs) Handles OnEditCrear_Exe.Click

txtEdit.Text = ""

System.Windows.Forms.Application.DoEvents()
End Sub
End Class
```

提案システムの実装

提案システムのプログラム

次ページからは提案システムのプログラムを記述している。ベースはサンプルプログラムを利用しているため、各関数の仕様などは基本的に変更していない。

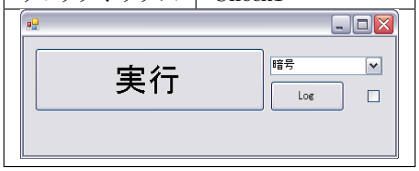
しかし、前述のサンプルプログラムの処理手順をそのまま使用した場合、カードを読み取る度に AID オブジェクトをセットアップするため、１回では成功できなかったときに、失敗を示す出力が表示されてしまう。そこで、フォームが開かれると同時に AID オブジェクトのセットアップを行い、以降はフォームが閉じられるまでは AID オブジェクトがセットアップ状態のままカードの読み書きができるよう構成を変更した。

フォームの仕様、プログラムリストは以降に示す。

提案システムのフォーム仕様

(a)FormMain

アイテム名	Name
フォーム	FormMain
ボタン 1	OnFeliCa_Exe
ボタン 2	LogBtn
コンボボックス	ComBox
チェックボックス	Check1

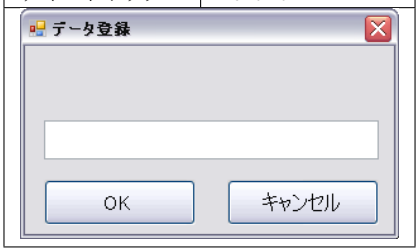


(b)FormSub

アイテム名	Name
フォーム	FormSub
ボタン	LogClear
テキストボックス	LogBox

(c)FormReg

アイテム名	Name
フォーム	FormReg
ボタン 1	RegBtn
ボタン 2	CanBtn
ラベル 1	Label1
ラベル 2	Label2
テキストボックス	TextBox



(d)FormEnc

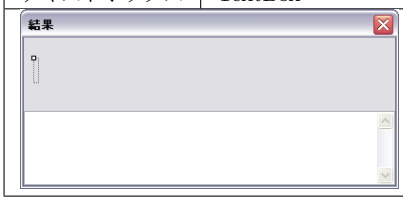
アイテム名	Name
フォーム	FormEnc
ボタン 1	OKBtn
ボタン 2	CancelBtn
ラベル	Label

(e)FormDec

アイテム名	Name
フォーム	FormDec
ボタン 1	OKBtn
ボタン 2	NextBtn
ボタン 3	CancelBtn
ラベル	Label

(f)FormResult

アイテム名	Name
フォーム	FormResult
ラベル	Label
テキストボックス	TextBox



メインフォームプログラム (FromMain.vb)

```
Option Strict Off
Option Explicit On
Imports System
Imports System.IO
Imports VB = Microsoft.VisualBasic

Friend Class FormMain
Inherits System.Windows.Forms.Form
    Private Shared Timer1 As New System.Windows.Forms.Timer()

'// 各オブジェクトの変数を宣言
Public ifd As Object '// IFD オブジェクト
Public icc As Object '// Card オブジェクト
Public fel As Object '// FeliCa オブジェクト

    Dim f As New FormSub
    Dim f2 As New FormReg
    Dim fEnc As New FormEnc
    Dim fDec As New FormDec
    Dim fResult As New FormResult

    Dim Temp As String
    Dim Flag As Boolean
    Private Shared Flag2 As Boolean = True
    Dim Flag3 As Short
'//=============================================================
    '// フォームがオープンされるときの動作
'//=============================================================
    Private Sub Form_Load(ByVal sender As System.Object, _
        ByVal e As System.EventArgs) Handles MyBase.Load
        Dim result As Integer
'//=============================================================
'// IFD/カード/API オブジェクトの生成例です
'//=============================================================
        upDateEdit_1("実行開始" & vbCrLf)
'//-------------------------------------------------------
'◆ Step1：IFD/カード/API オブジェクトの生成
```

```
            If result = NORMALEND Then
                result = CreateObject_Ex()
            End If


'//-------------------------------------------------
'◆ Step2：IFD/カード/API オブジェクトに対する SetUp

            If result = NORMALEND Then
                Do
                    Application.DoEvents()
                    result = ObjectSetUp_Ex()
                Loop While (result)
            End If
            '// キャリア出力ウェイト時間変更
            If result = NORMALEND Then
                result = SetCarrierWaitTime_Ex()
            End If

'//-------------------------------------------------
'◆ Step3：ホストとリーダ/ライタ間の通信速度を
'   115200bps へ変更（デフォルトは 9600bps です）
'   変更不要な場合、この処理は削除可能です

            If result = NORMALEND Then
                result = ChangeSpeed_Ex(DEF_115200)
            End If


'//-------------------------------------------------
'◆ Step4：ホストとリーダ/ライタ間の相互認証

            If result = NORMALEND Then
                result = RW_Authentication_Ex()
            End If

        End Sub


'//=============================================================
```

```
'// AIDを使った一連の手順例
'//=============================================================
Private Sub OnFeliCa_Exe_Click(ByVal eventSender _
                As System.Object, ByVal eventArgs _
                  As System.EventArgs) Handles OnFeliCa_Exe.Click

        f.Owner = Me
        f2.Owner = Me
        fEnc.Owner = Me
        fDec.Owner = Me
        fResult.Owner = Me

        OnFeliCa_Exe.Text = "中止"

        RemoveHandler OnFeliCa_Exe.Click, _
                          AddressOf OnFeliCa_Exe_Click
        AddHandler OnFeliCa_Exe.Click, AddressOf StBtn_Click
        MyBase.ControlBox = False


'//-----------------------------------------------------
            '   暗号・複合モードの選択値による動作
            Select Case Flag3
                Case 1
                    Call upDateEdit_6(Encrypt())

                Case 2
                    Dim Res As String
                    Res = Decrypt()
                    fResult.TextBox.Text = Res
                    If Res <> "処理がキャンセルされました" Then
                      fResult.Label.Text = "結果は以下の通りです"
                        fResult.ShowDialog()
                    Else
                        Call upDateEdit_6(Res)
                    End If
                 Case Else

            End Select
```

```
        Call upDateEdit_1("停止" & vbCrLf)

      AddHandler OnFeliCa_Exe.Click, AddressOf OnFeliCa_Exe_Click
      RemoveHandler OnFeliCa_Exe.Click, AddressOf StBtn_Click
        OnFeliCa_Exe.Text = "実行"
        Label.Text = ""
        MyBase.ControlBox = True

    End Sub

'//=============================================================
    '// フォームが閉じられるときの動作
'//=============================================================
    Private Sub Form_Closing(ByVal sender As System.Object, _
        ByVal e As System.EventArgs) Handles MyBase.FormClosing
'//-------------------------------------------------
'◆ Step10：IFD/カード/API オブジェクトの開放
        Call ReleaseAid_Ex()
        Call upDateEdit_1("終了しています...")
        OnFeliCa_Exe.Text = "終了中"
        Call upDateEdit_6("終了しています...")
    End Sub

'//=============================================================
    '// 暗号データ入力部
'//=============================================================
    Private Function Encrypt() As String
        Dim MFlag As Boolean = 0
        Dim key As Byte()
        Dim Res As String = "処理がキャンセルされました"
        f2.Label1.Text = "登録するキーを入力します"
        f2.Label2.Text = "※半角英数で入力してください"
        If f2.ShowDialog(Me) = DialogResult.OK Then
            key = System.Text.Encoding.UTF8.GetBytes( _
                                        f2.TextBox.Text)
            upDateEdit_1("秘密情報=>")
            HexOutput(key)
            f2.TextBox.Text = ""
        Else
```

```
    Return (Res)
    Exit Function
End If
f2.Label1.Text = "登録するカードの"
f2.Label2.Text = "最大枚数を入力してください"
Dim CardNum As Integer
If f2.ShowDialog(Me) = DialogResult.OK Then
    CardNum = CInt(f2.TextBox.Text)
    f2.TextBox.Text = ""
Else
    Return (Res)
    Exit Function
End If
f2.Label1.Text = "カードの認証可能な"
f2.Label2.Text = "枚数を入力してください"
Dim CardNumK As Integer
If f2.ShowDialog(Me) = DialogResult.OK Then
    CardNumK = CInt(f2.TextBox.Text)
    f2.TextBox.Text = ""
Else
    Return (Res)
    Exit Function
End If

'   暗号化本体
Dim Rnd As New Random
Dim a As Integer = Rnd.Next(100, 500)
Dim p As Integer = 2147483647
'   Prime = 2147483647
Dim IDm(CardNum) As String
Dim DataInt(key.Length - 1) As Integer
Dim DataByte(31) As Byte
Dim data As Integer
Dim m As Integer
Dim n As Integer
Dim count As Integer = 0
Dim count2 As Integer
Dim y As Integer
Dim result As Integer
For m = 1 To CardNum
```

```
            fEnc.Label.Text = m & "枚目のカードをかざしてください"
            Call upDateEdit_1("Card No." & m)
            If fEnc.ShowDialog(Me) = DialogResult.OK Then
                Do
                    Application.DoEvents()
'//-------------------------------------------------------
                    '◆ StartTransaction を実行しを捕捉
                    result = CatchCard_Ex()
                    If result = NORMALEND Then
                        Exit Do
                    End If
                Loop While (1)
                Label.Text = "カードを動かさないでください"
                count2 = 0
                For n = 0 To key.Length - 1
                    data = SSS(m, a, key(n), CardNumK, p)
                    y = 1
                    Do
                        If count = 0 Then
                            DataByte(0) = m
                            DataByte(16) = &HFF
                            count = 1
                        End If
                        DataByte(count) = CByte(data Mod 256)
                        data = data \ 256
                        DataByte(count + 16) = y
                        count += 1
                        y += 1
                        If count = 16 Then
                            result = Fel_Write_Ex(DataByte, _
                                                  count2)
                            HexOutput(DataByte)
                            count = 0
                            count2 += 2
                        ElseIf data = 0 And n = _
                                  (key.Length - 1) Then
                            For count = count To 15
                                DataByte(count) = 0
                                DataByte(count + 16) = 0
                            Next
```

```
                            result = Fel_Write_Ex(DataByte, _
                                                  count2)
                            HexOutput(DataByte)
                            If result = NORMALEND Then
                                For count = 0 To 31
                                    DataByte(count) = 0
                                Next count
                                count = 0
                                result = Fel_Write_Ex(DataByte, _
                                                      count2 + 2)
                                result = ReleaseCard_Ex()
                            End If
                        End If
                    Loop While (data)
                Next n

            Else
                Return (Res)
                Exit Function
            End If
        Next m
        Res = "処理が正常に完了しました"

        Return (Res)
    End Function
'//=============================================================
    '// 暗号処理本体
'//=============================================================
    Private Function SSS(ByVal x As Integer, ByVal a As _
        Integer, ByVal s As Integer, ByVal k As Integer, _
                    ByVal p As Integer)
        Dim answer As Integer = s
        Dim j As Integer
        For j = 1 To (k - 1)
            answer = answer + (a - 1 + j) * ((x ^ (k - 1)))
        Next j
        Return (answer)
    End Function
'//=============================================================
    '// 復号データ入力部
```

```
'//=============================================================
    Private Function Decrypt() As String
        Dim Res As String = "処理がキャンセルされました"
        Dim tmpData(34) As Byte
        Dim tmpData2() As Byte
        Dim tmpInt(2) As Integer
        Dim m As Integer = 1
        Dim j As Integer = 0
        Dim q As Integer
        Dim result As Integer
        Dim Data As New ArrayList
        Dim ObjCount As Integer = 0
        Dim p As Integer


        fDec.Label.Text = m & "枚目のカードをかざしてください"
        Do While ((fDec.ShowDialog(Me) <> DialogResult.Cancel))
            Do
                Label.Text = "カードを動かさないでください"
                Application.DoEvents()
'//-------------------------------------------------------------
                '◆ StartTransaction を実行しを捕捉
                result = CatchCard_Ex()
                If result = NORMALEND Then
                    Exit Do
                End If
            Loop While (1)
            upDateEdit_1(m & "枚目捕捉完了")
            Do
                tmpData2 = Fel_Read_Ex(j)
                For q = 0 To 33
                    tmpData(q) = tmpData2(q)
                Next q
                If (tmpData(0) = 0 And tmpData(1) = 0 And _
                     tmpData(19) = &HFF) Then

                    upDateEdit_1(m & "枚目  " & j * 2 & "+" & _
                          j * 2 + 1 & "ブロックデータ")
                    upDateEdit_1("データの長さ" & _
                          tmpData.Length - 1 & "バイト")
```

```
HexOutput(tmpData)

For p = 20 To 34
    If p = 20 Then
        If tmpData(p) = 1 Then
            If j > 0 Then
                Call upDateEdit_1("Data" & _
              ObjCount & "=" & tmpInt(2))
                Data.Add(tmpInt.Clone)
                ObjCount += 1
            End If
            tmpInt(0) = tmpData(3)
            tmpInt(1) = ObjCount
            tmpInt(2) = CInt(tmpData(4)) * _
            (256 ^ (CInt(tmpData(20)) - 1))
        Else
            tmpInt(0) = tmpData(3)
            tmpInt(1) = ObjCount
            tmpInt(2) = tmpInt(2) + _
       CInt(tmpData(4)) * (256 ^ _
         (CInt(tmpData(20)) - 1))
        End If
    ElseIf tmpData(p) <> 0 And tmpData( _
                          p - 1) _
                < tmpData(p) Then
        tmpInt(2) = tmpInt(2) + _
            CInt(tmpData(p - 16)) * _
           (256 ^ (CInt(tmpData(p)) - 1))
    ElseIf tmpData(p) <> 0 And tmpData( _
              p - 1) >= tmpData(p) Then
        Data.Add(tmpInt.Clone)
        Call upDateEdit_1("Data" & _
            ObjCount & "=" & tmpInt(2))
        ObjCount += 1
        tmpInt(0) = tmpData(3)
        tmpInt(1) = ObjCount
        tmpInt(2) = CInt(tmpData(p - 16)) * _
           (256 ^ (CInt(tmpData(p)) - 1))
    ElseIf tmpData(p) = 0 Then
        Data.Add(tmpInt.Clone)
```

```
                            Call upDateEdit_1("Data" & _
                                ObjCount & "=" & tmpInt(2))
                            Exit For
                        End If
                    Next p
                Else
                    Exit Do
                End If
                j += 1
            Loop While (1)
            j = 0
            If fDec.DialogResult = Windows.Forms. _
                        DialogResult.OK Then
                Exit Do
            End If
            m += 1
            ObjCount = 0
            fDec.Label.Text = m & "枚目のカードをかざしてください"
            Label.Text = ""
        Loop
        If fDec.DialogResult = Windows.Forms.DialogResult _
                                            .Cancel Then
            Return (Res)
            Exit Function
        End If

        Res = DeSSS(Data, m)

        Call upDateEdit_1("復号結果=" & Res)
        Return (Res)
    End Function
'//=============================================================
    '// 復号処理本体
'//=============================================================
    Private Function DeSSS(ByVal Data As ArrayList, _
                    ByVal k As Integer) As String
        Dim counter As Integer
        Dim size As Integer = Data.Count / k
        Dim i As Integer
        Dim j As Integer
```

```
        Dim ij(k - 1) As Integer
        Dim F(k - 1) As Integer

        Dim ResStr As String = ""
        Dim answer As Double = 0
        Dim Lam As Double = 1
        Dim LamA As Double = 1
        Dim LamB As Double = 1
        Dim LamS As Double

        For counter = 0 To size - 1

            For i = 0 To k - 1
                ij(i) = Data((i * size) + counter)(0)
                F(i) = Data((i * size) + counter)(2)
            Next i

            For i = 1 To k
                For j = 1 To k
                    If j <> i Then
                        LamA = LamA * (-ij(j - 1))
                        LamB = LamB * (ij(i - 1) - ij(j - 1))
                        Lam = Lam * (-ij(j - 1)) / (ij(i - 1) _
                              - ij(j - 1))
                    End If
                Next j
                LamS = LamA / LamB
                answer = answer + LamS * F(i - 1)
                Lam = 1
                LamA = 1
                LamB = 1
            Next i

            ResStr = ResStr + CStr(Chr(CInt(answer)))
            answer = 0
        Next counter


        Return (ResStr)
    End Function
```

```
'//=============================================================
'// IFD/カード/API オブジェクトの生成例です
'//=============================================================
Private Function CreateObject_Ex() As Short

On Error Resume Next

'// オブジェクトが既に存在する場合は開放
If ((ifd Is Nothing) = False) Or ((icc Is Nothing) _
                  = False) Or ((fel Is Nothing) = False) Then
Call ReleaseAid_Ex()
End If


'//-------------------------------------------------------------
'// IFD オブジェクトの生成
ifd = CreateObject("IFD.RW")
If (ifd Is Nothing) <> False Then
Call upDateEdit_1("IFD Object Create Error")
CreateObject_Ex = 1
Exit Function
End If
Call upDateEdit_1("IFD Object Create OK")


'//-------------------------------------------------------------
'// カードオブジェクトの生成
icc = CreateObject("ICC.Card")
If (icc Is Nothing) <> False Then
Call upDateEdit_1("Card Object Create Error")
CreateObject_Ex = 1
Exit Function
End If
Call upDateEdit_1("Card Object Create OK")


'//-------------------------------------------------------------
```

```
'// FeliCa APIオブジェクトの生成
fel = CreateObject("FeliCa2.API")
If (fel Is Nothing) <> False Then
Call upDateEdit_1("FeliCa API Object Create Error")
CreateObject_Ex = 1
Exit Function
End If
Call upDateEdit_1("FeliCa API Object Create OK")

CreateObject_Ex = NORMALEND

End Function

'//=============================================================
'// IFD/カード/APIオブジェクトに対するSetUp例です
'// (PR-400/500用機能使用通知を含みます)
'//=============================================================
Private Function ObjectSetUp_Ex() As Integer

Dim result As Integer
Dim portName As String
Dim i As Short


'//-------------------------------------------------------------
'◆ IFDオブジェクトのSetUp
'   COM1～COM9まで順番に接続していきます
'(COM10以上接続時は、変更が必要です)

For i = 1 To 9
portName = "COM" & CStr(i)
result = ifd.SetUp("PR300", portName)
If result = NORMALEND Then
Exit For
End If
Next i
If result <> NORMALEND Then
Call upDateEdit_2("IFD Object Setup Error : ", result)
ObjectSetUp_Ex = result
Exit Function
```

```
End If
Call upDateEdit_2("IFD Object Setup OK : ", result)
Call upDateEdit_1("  Open Port : " & portName)


'//-------------------------------------------------------------
'◆ PR-400/500 用機能使用通知
'   (チェックボックスのチェック有無により実行可否を判定)
'   本例では、この処理を実行しなくても動作可能です
'   (PR-301 使用時はを外してください)

If Check1.CheckState = 1 Then
result = SetPR400_500FuncUse_Ex()
If result <> NORMALEND Then
ObjectSetUp_Ex = result
Exit Function
End If
End If


'//-------------------------------------------------------------
'◆ カードオブジェクトの SetUp

result = icc.SetUp("CF:FEL2", ifd)
If result <> NORMALEND Then
Call upDateEdit_2("Card Object SetUp Error : ", result)
ObjectSetUp_Ex = result
Exit Function
End If
Call upDateEdit_2("Card Object SetUp OK : ", result)


'//-------------------------------------------------------------
'◆ API オブジェクトの SetUp

result = fel.SetUp(icc)
If result <> NORMALEND Then
Call upDateEdit_2("FeliCa API Object SetUp Error : ", result)
ObjectSetUp_Ex = result
Exit Function
```

```
End If
Call upDateEdit_2("FeliCa API Object SetUp OK : ", result)

ObjectSetUp_Ex = result

End Function

'//============================================================
'// IFD/カード/API オブジェクトの開放例です
'//============================================================
Private Sub ReleaseAid_Ex()

'// FeliCa API オブジェクトの開放
If (fel Is Nothing) = False Then
fel = Nothing '// Release API オブジェクト
End If

'// カードオブジェクトの開放
If (icc Is Nothing) = False Then
icc = Nothing '// Release カードオブジェクト
End If

'// IFD オブジェクトの開放
If (ifd Is Nothing) = False Then
ifd = Nothing '// Release IFD オブジェクト
End If

Call upDateEdit_1("Release All Aid Object OK")

End Sub

'//============================================================
'// PR-400/500 機能使用通知例です
'//============================================================
Private Function SetPR400_500FuncUse_Ex() As Integer

Dim result As Integer
Dim vtag As Integer
Dim vvalue(256) As Byte
Dim vinfBuff(256) As Byte
```

```
Dim vbuffSize As Integer
Dim vinfLen(1) As Integer


vtag = &H4080500 '// 制御タグ値
vvalue(0) = &H0 '// リーダ/ライタ情報書き込み命令
vvalue(1) = &H1 '// PR-400/PR-500用機能を使用

vbuffSize = 256

result = ifd.SetControl(vtag, vvalue, vinfBuff, _
                                          vbuffSize, vinfLen)

If result <> NORMALEND Then
Call upDateEdit_2("PR-400/500 Function Set Error : ", result)
Else
Call upDateEdit_2("PR-400/500 Function Set OK : ", result)
End If

SetPR400_500FuncUse_Ex = result

End Function

'//============================================================
'// ホストとリーダ/ライタ間の通信速度の変更例です
'//============================================================
Private Function ChangeSpeed_Ex(ByVal SetSpeed As Integer) _
                                                  As Integer

Dim result As Integer
Dim vtag As Integer
Dim vvalue(256) As Integer
Dim vinfBuff(256) As Byte
Dim vbuffSize As Integer
Dim vinfLen(1) As Integer


vtag = &H4010007 '// 制御タグ値
vvalue(0) = SetSpeed '// 設定する通信速度：引数で指定(bps)
```

```
vbuffSize = 256

result = ifd.SetControl(vtag, vvalue, vinfBuff, _
                                              vbuffSize, vinfLen)

If result <> NORMALEND Then
If SetSpeed = DEF_9600 Then
Call upDateEdit_2("Change Baudrate 9600 Error : ", result)
Else
Call upDateEdit_2("Change Baudrate 115200 Error : ", result)
End If
Else
If SetSpeed = DEF_9600 Then
Call upDateEdit_2("Change Baudrate 9600 OK : ", result)
Else
Call upDateEdit_2("Change Baudrate 115200 OK : ", result)
End If
End If

ChangeSpeed_Ex = result

End Function

'//============================================================
'// ホストとリーダ/ライタ間での相互認証例です
'//============================================================
Private Function RW_Authentication_Ex() As Integer

Dim result As Integer
Dim vtag As Integer
Dim vvalue(256) As Byte
Dim vinfBuff(256) As Byte
Dim vbuffSize As Integer
Dim vinfLen(1) As Integer


'//------------------------------------------------------------
'◆ IFD オブジェクトにリーダ/ライタ側の相互認証鍵と
'   暗号モードを設定しますリーダ/ライタの相互認証鍵、
'   暗号モードが初期状態の場合の処理例です
```

```
vtag = &H4088010 '// 制御タグ値

vvalue(0) = &H3 '// 初期化設定条件

vvalue(1) = &HFF '// 現在のリーダ/ライタの相互認証鍵１（初期値）
vvalue(2) = &HFF
vvalue(3) = &HFF
vvalue(4) = &HFF
vvalue(5) = &HFF
vvalue(6) = &HFF
vvalue(7) = &HFF
vvalue(8) = &HFF

vvalue(9) = &HFF '// 現在のリーダ/ライタの相互認証鍵２（初期値）
vvalue(10) = &HFF
vvalue(11) = &HFF
vvalue(12) = &HFF
vvalue(13) = &HFF
vvalue(14) = &HFF
vvalue(15) = &HFF
vvalue(16) = &HFF

vvalue(17) = &HFF '// 現在のリーダ/ライタの相互認証鍵３（初期値）
vvalue(18) = &HFF
vvalue(19) = &HFF
vvalue(20) = &HFF
vvalue(21) = &HFF
vvalue(22) = &HFF
vvalue(23) = &HFF
vvalue(24) = &HFF

vvalue(25) = &H0 '// 認証アルゴリズム：適用なし
vvalue(26) = &H0 '// 暗号通信機能：適用なし
vvalue(27) = &H0 '// 認証暗号設定情報の暗号化：適用なし

vbuffSize = 256

result = ifd.SetControl(vtag, vvalue, vinfBuff, _
                                          vbuffSize, vinfLen)
```

```
If result <> NORMALEND Then
Call upDateEdit_2("IFD Authentication Set Error : ", result)
RW_Authentication_Ex = result
Exit Function
End If
Call upDateEdit_2("IFD Authentication Set OK : ", result)


System.Array.Clear(vvalue, 0, vvalue.Length)
System.Array.Clear(vinfLen, 0, vinfLen.Length)


'//-------------------------------------------------------------
'◆ IFD オブジェクトで、リーダ/ライタと相互認証 1・2 を実行

vtag = &H4088003 '// 制御タグ値
vvalue(0) = 0

result = ifd.SetControl(vtag, vvalue, vinfBuff, _
                                          vbuffSize, vinfLen)

If result <> NORMALEND Then
Call upDateEdit_2("R/W Authentication Error : ", result)
Else
Call upDateEdit_2("R/W Authentication OK : ", result)
End If

RW_Authentication_Ex = result

End Function

'//=============================================================
'// StartTransaction を実行しカードを捕捉する例です
'//=============================================================
Private Function CatchCard_Ex() As Integer

Dim result As Integer
Dim vmode As Integer
Dim vspeed As Integer
```

```
Dim vtype(1) As Integer
Dim vinfBuff(256) As Byte
Dim vbuffSize As Integer
Dim vinfLen(1) As Integer


vmode = 1000 '// 検出タイムアウト時間を 1000ms とします
vspeed = CARDSPEED_DEFAULT '// カードとの初期通信速度
vtype(0) = 0 '// カード検出タイプ

vbuffSize = 256

'// カードオブジェクトを使用した StartTransaction の実行
result = icc.StartTransaction(vmode, vspeed, vtype, _
                        vinfBuff, vbuffSize, vinfLen)

        If result <> NORMALEND Then
            Call upDateEdit_1(" トランザクションエラー")
            Call upDateEdit_2("StartTransaction Error : ", _
                                                result)
            CatchCard_Ex = result
            Exit Function
        End If
Call upDateEdit_2("StartTransaction OK : ", result)


'// 捕捉したカードの処理例

'// FeliCa カードの捕捉に成功
If (vtype(0) And (Not CARDTYPE_MORECARD)) = CARDTYPE_C Then
'// FeliCa カード捕捉時の処理を記述します
Call upDateEdit_1("  Catch FeliCa Card")

'// FeliCa カード以外のカードを捕捉（通常ない）
Else
'// FeliCa 以外のカード捕捉時の処理を記述します
Call upDateEdit_1("  Catch Another Card")

End If
        Call upDateEdit_3(vinfBuff, vinfLen(0))
```

```
'// 複数カード捕捉
If (vtype(0) And CARDTYPE_MORECARD) Then
'// 他のカード捕捉時の処理を記述します
Call upDateEdit_1("  More Card Exist")

End If

CatchCard_Ex = result

End Function

'//============================================================
'// ReadWithoutEncryptionを実行しカード内のデータを読み出す例です
'//============================================================
    Private Function Fel_Read_Ex(ByVal num As Integer) As Byte()

        Dim result As Integer
        Dim vsCount As Byte
        Dim vsList(1) As Byte
        Dim vbCount As Byte
        Dim vbList(3) As Byte
        Dim vrxBuff(256) As Byte
        Dim vbuffSize As Integer
        Dim vrxLen(1) As Integer

        Dim data(31) As Byte


        vsCount = &H1
        vsList(0) = &H9
        vsList(1) = &H10

        num *= 2

        vbCount = &H2
        vbList(0) = &H80
        vbList(1) = num
        vbList(2) = &H80
        vbList(3) = num + 1
```

```
        vbuffSize = 256

        If num > 10 Then
            vsList(0) = &H9
            vsList(1) = &H12

            vbList(1) = (num - 12)
            vbList(3) = (num - 11)
        End If

        result = fel.ReadWithoutEncryption(vsCount, vsList, _
               vbCount, vbList, vrxBuff, vbuffSize, vrxLen)

        If result <> NORMALEND Then
            Call upDateEdit_6("カード読み取り失敗")
            Fel_Read_Ex = data
            Exit Function
        End If

        '// ReadWithoutEncryption にて取得したの解釈
        If (vrxLen(0) < 2) Or (vrxBuff(0) <> &H0) Or _
                  (vrxBuff(1) <> &H0) Then
            Call upDateEdit_6("カードデータエラー")
            Label.Text = Label.Text & vbCrLf & vbCrLf & _
                   vbCrLf & "カードをかざしてください"
            Fel_Read_Ex = data
            Exit Function
        End If
        Fel_Read_Ex = vrxBuff

    End Function
'//=============================================================
    '// WriteWithoutEncryption を実行し
    '//カード内のデータを読み出す例です
'//=============================================================
    Private Function Fel_Write_Ex(ByVal Data As Byte(), _
             ByVal num As Integer) As Integer

        Dim result As Integer
```

```
Dim vsCount As Byte
Dim vsList(1) As Byte
Dim vbCount As Byte
Dim vbList(3) As Byte
Dim vbData(31) As Byte
Dim vrxBuff(256) As Byte
Dim vbuffSize As Integer
Dim vrxLen(1) As Integer

vsCount = &H1
vsList(0) = &H9
vsList(1) = &H10

vbCount = &H2
vbList(0) = &H80
vbList(1) = num
vbList(2) = &H80
vbList(3) = num + 1

If num > 10 Then
    vsList(0) = &H9
    vsList(1) = &H12

    vbList(1) = (num - 12)
    vbList(3) = (num - 11)
End If

vbData = Data
vbuffSize = 256

result = fel.WriteWithoutEncryption(vsCount, vsList, _
   vbCount, vbList, vbData, vrxBuff, vbuffSize, vrxLen)

If result <> NORMALEND Then
    Call upDateEdit_2("WriteWithoutEncryption Command _
              Error : ", result)
    Fel_Write_Ex = result
    Exit Function
End If
```

```
        '// ReadWithoutEncryption にて取得したの解釈
        If (vrxLen(0) >= 2) And (vrxBuff(0) = &H0) And _
                    (vrxBuff(1) = &H0) Then

        Else
            Call upDateEdit_1("  WriteWithoutEncryption Data _
                      Contents Error")
        End If

        Fel_Write_Ex = result

    End Function



'//=============================================================
'// EndTransaction を実行しカードを開放する例です
'//=============================================================
Private Function ReleaseCard_Ex() As Integer

Dim result As Integer
Dim vmode As Integer


vmode = 0 '// 開放条件：モード 0

'// カードオブジェクトを使用した EndTransaction の実行
result = icc.EndTransaction(vmode)

If result <> NORMALEND Then
Call upDateEdit_2("EndTransaction Error : ", result)
Else
Call upDateEdit_2("EndTransaction OK : ", result)
End If

ReleaseCard_Ex = result

End Function

'//=============================================================
```

```
'// Edit へのコメント、result、Data を出力
'//=============================================================
    Private Sub upDateEdit_1(ByVal OutMsg As String)
        f.LogBox.Text = f.LogBox.Text & OutMsg & vbCrLf
        f.LogBox.SelectionStart = Len(f.LogBox.Text)
        f.LogBox.ScrollToCaret()

        System.Windows.Forms.Application.DoEvents()

    End Sub

Private Sub upDateEdit_2(ByVal OutMsg As String, _
                                    ByVal result As Integer)

txtEdit.Text = txtEdit.Text & OutMsg & "[Result] " & _
              VB6.Format(Hex(result), "00000000") & vbCrLf
txtEdit.SelectionStart = Len(txtEdit.Text)

System.Windows.Forms.Application.DoEvents()

End Sub

    End Sub
    Private Sub upDateEdit_4(ByVal result As Integer)

        txtEdit.Text = txtEdit.Text & "[Result] の内容=> _
            " & VB6.Format(Hex(result), "00000000") & vbCrLf
        txtEdit.SelectionStart = Len(txtEdit.Text)

        System.Windows.Forms.Application.DoEvents()

    End Sub

    Private Sub upDateEdit_6(ByVal OutMsg As String)
        Timer1.Interval = 2500
        AddHandler Timer1.Tick, AddressOf TimerEventProcessor
        Timer1.Start()
        Do While (Flag2)
            Label.Text = OutMsg
            OnFeliCa_Exe.Enabled = False
```

```
            System.Windows.Forms.Application.DoEvents()
        Loop
        Flag2 = True
        OnFeliCa_Exe.Enabled = True
        RemoveHandler Timer1.Tick, AddressOf TimerEventProcessor
    End Sub

    Private Sub HexOutput(ByVal data As Byte())
        Dim max As Integer = data.Length - 1
        Dim i As Short
        Dim str As String = ""
        For i = 0 To max
            str = str & VB.Right("0" & Hex(data(i)), 2) & " "
        Next
        Call upDateEdit_1(str)

    End Sub

    Private Shared Sub TimerEventProcessor(ByVal myObject _
          As Object, ByVal myEventArgs As EventArgs)
        Timer1.Stop()
        Flag2 = False
    End Sub

    Private Sub StBtn_Click(ByVal sender As System.Object, _
             ByVal e As System.EventArgs)
        Flag = True
    End Sub

    Dim LFlag As Boolean

    Private Sub LogBtn_Click(ByVal sender As System.Object, _
         ByVal e As System.EventArgs) Handles LogBtn.Click
        If LFlag = False Then
            f.Show()
            LFlag = True
        Else
            f.Hide()
            LFlag = False
        End If
```

```
    End Sub

    Private Sub ComBox_SelectedIndexChanged(ByVal sender As _
        System.Object, ByVal e As System.EventArgs) _
              Handles ComBox.SelectedIndexChanged
        Select Case ComBox.Text
            Case "暗号"
                Flag3 = 1
            Case "復号"
                Flag3 = 2
            Case Else
                Flag3 = 0
        End Select

    End Sub
End Class
```

サブフォームプログラム (FormSub.vb)

```
Public Class FormSub
    Private Sub LogClear_Click(ByVal sender As System.Object, _
      ByVal e As System.EventArgs) Handles LogClear.Click
        LogBox.Text = ""
        System.Windows.Forms.Application.DoEvents()
    End Sub
End Class
```

サブフォームプログラム (FormReg.vb)

```
Public Class FormReg
    Private Sub RegBtn_Click(ByVal sender As System.Object, _
      ByVal e As System.EventArgs) Handles RegBtn.Click
        If TextBox.Text = "" Then
            Label1.Text = "テキストボックスに"
            Label1.BackColor = Color.Red
            Label2.Text = "名前を入力してください。"
            Label2.BackColor = Color.Red
            Exit Sub
        End If
        Me.DialogResult = DialogResult.OK
        Me.Close()
```

```
    End Sub

    Private Sub CanBtn_Click(ByVal sender As System.Object, _
      ByVal e As System.EventArgs) Handles CanBtn.Click
        Me.DialogResult = DialogResult.Cancel
        Me.Close()
    End Sub
End Class
```

サブフォームプログラム (FormEnc.vb)

```
Public Class FormEnc
    Private Sub Button_Click(ByVal sender As System.Object, _
      ByVal e As System.EventArgs) Handles OKBtn.Click
        Me.DialogResult = DialogResult.OK
        Me.Close()
        Label.Text = ""
    End Sub

    Private Sub CancelBtn_Click(ByVal sender As System.Object, _
      ByVal e As System.EventArgs) Handles CancelBtn.Click
        Me.DialogResult = DialogResult.Cancel
        Me.Close()
        Label.Text = ""
    End Sub
End Class
```

サブフォームプログラム (FormDec.vb)

```
Public Class FormDec
    Private Sub OKBtn_Click(ByVal sender As System.Object, _
      ByVal e As System.EventArgs) Handles OKBtn.Click
        Me.DialogResult = DialogResult.OK
        Me.Close()
        Label.Text = ""

    End Sub

    Private Sub NextBtn_Click(ByVal sender As System.Object, _
      ByVal e As System.EventArgs) Handles NextBtn.Click
        Me.DialogResult = DialogResult.Retry
        Me.Close()
```

```
        Label.Text = ""

    End Sub

    Private Sub CancelBtn_Click(ByVal sender As System.Object, _
      ByVal e As System.EventArgs) Handles CancelBtn.Click
        Me.DialogResult = DialogResult.Cancel
        Me.Close()
        Label.Text = ""
    End Sub
End Class
```

サブフォームプログラム (FormResult.vb)

```
Public Class FormResult
    Private Sub FormResult_Load(ByVal sender As System.Object, _
      ByVal e As System.EventArgs) Handles MyBase.Click
        Me.DialogResult = DialogResult.No
        Me.Close()
    End Sub
End Class
```

質疑応答

1. Q：この提案システムで推奨される、認証可能数はどのくらいですか？
（能登教授）
A：分散させる秘密情報の容量によって異なりますが、３～４が推奨数です。

2. Q：FeliCa のカードに保存できる容量はどのくらいですか？
（豊嶋教授）
A：製品によって保存容量は違いますが、9K バイトの FRAM が搭載されているものがあります。
また、フォーマットによって利用可能なブロックが決められるため、保存容量もそれに応じて変化します。